A Security Architecture for a Peer-to-Peer Video Conference System

Von der Fakultät für Mathematik, Naturwissenschaft und Informatik
der Brandenburgischen Technischen Universität Cottbus

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

genehmigte Dissertation
vorgelegt von

Diplom-Ingenieur

Fuwen Liu

Geboren am 9. Juli 1967 in Sichuan (V. R. China)

Gutachter: Prof. Dr.-Ing. Hartmut König
Gutachter: Prof. Dr. rer. nat. Peter Herrmann
Gutachter: Prof. Dr.-Ing. Rolf Kraemer

Tag der mündlichen Prüfung: 04.12.2006

Abstract

More and more modern group oriented collaborative applications use the peer-to-peer (P2P) paradigm to be independent of expensive infrastructures as they are, for instance, provided for audio and video conferences by H.323 systems. Decentralized P2P systems better support spontaneity and mobility to set up meetings at varying locations or in ad hoc environments. This is especially advantageous for business communication over the Internet but also for other collaborative applications such as audio/video conferences.

Decentralized collaborative P2P solutions require appropriate mechanisms to protect group privacy and data integrity. The broadly available security infrastructures, like virtual private networks (VPN) in the link, network, and transport layer, do not well fulfill the requirements regarding security, efficiency, and flexibility raised by a conferencing system. A dedicated security architecture in the application layer is, therefore, highly desired for protecting a P2P video conference. A centralized client/server based video conference system can be well shielded in a standard manner, whilst there exist no off-the-shelf approaches to specifying how to secure a P2P video conference up to now.

This Ph.D. thesis addresses this urgent issue by presenting an effective and flexible security architecture and showing how it can be embedded into a P2P video conferencing system using the BRAVIS system as example. The cornerstone of the security architecture is the decentralized group key management. For this purpose, a new distributed key exchange protocol has been proposed. It is especially well-suited for applications in real-time P2P settings for its higher efficiency than existing ones concerning the group key renewal delay. Furthermore, a novel video encryption algorithm has been developed to meet the strict real-time constraints required in a video conference. Its outstanding features include a good balance between security and efficiency, no impairment on video compression efficiency, and readily integration into the existing multimedia systems. These make it more practicable to encrypt video data than existing approaches.

Zusammenfassung

Mehr und mehr moderne gruppenorientierte und kollaborative Applikationen nutzen das peer-to-peer (P2P)-Prinzip. Es bietet gegenüber dem zentralistischen Client/Server-Ansatz den Vorteil einer größeren Unabhängigkeit von einer möglicherweise teuren Infrastruktur, wie sie z. B. für Audio- und Videokonferenzen nach den H.32x-Standards erforderlich ist. Dezentrale P2P-Lösungen unterstützen besser die Spontaneität und Mobilität der Nutzer. Sie erlauben es, Sitzungen an unterschiedlichen Orten oder in Ad hoc-Umgebungen abzuhalten. Dies ist insbesondere für die geschäftliche Kommunikation über das Internet aber auch für andere kollaborative Anwendungen wie Audio/Video-Konferenzen von Vorteil.

Dezentralisierte kollaborative P2P-Lösungen erfordern geeignete Mechanismen, um die Privatheit der Gruppen und Integrität der Daten abzusichern. Vorhandene Sicherheitsinfrastrukturen wie Virtuelle Private Netze (VPN) erfüllen nicht ausreichend die Anforderungen bezüglich Sicherheit, Effizienz und Flexibilität, die an ein Konferenzsystem gestellt werden. Eine anwendungsbezogene Sicherheitsarchitektur ist daher in hohem Maße für den Schutz von P2P-Videokonferenzen wünschenswert. Für den Schutz zentralisierter Client/Server-Videokonferenzsysteme existieren bereits standardisierte Lösungen, während es für P2P-Videokonferenz noch keine Standardverfahren gibt.

Die vorliegende Dissertationsschrift widmet sich diesem wichtigen Thema ein. Sie stellt eine wirkungsvolle und flexible Sicherheitsarchitektur vor und zeigt am Beispiel des Videokonferenzsystems BRAVIS, wie sie in ein P2P-Videokonferenzsystem eingebettet werden kann. Grundstein der Sicherheitsarchitektur ist das dezentralisierte Gruppenschlüsselmanagement. Hierfür wird ein neues verteiltes Schlüsselaustauschprotokoll vorgeschlagen, das besonders den Anforderungen von Realzeit-P2P-Anwendungen angepasst wurde. Es besitzt eine höhere Effizienz als existierende Schlüsselaustauschprotokolle für die Gruppenschlüsselerneuerung. Weiterhin wird ein neuartiger Videoverschlüsselungs-Algorithmus vorgestellt, der den strengen Realzeitbedingungen einer Videokonferenz gerecht wird. Seine hervorzuhebenden Merkmale sind ein guter Abgleich zwischen Sicherheit und Leistungsfähigkeit, keine Beeinträchtigung der Videokompressionsleistung und einfache Integration in vorhandene Multimediasysteme. Diese Eigenschaften machen ihn praktikabler für das Verschlüsseln von Videodaten als existierende Ansätze.

Acknowledgements

I would like to utilize this opportunity to express my sincere acknowledgements to the people who provided support to my doctoral work in the past five years.

First of all, I would especially thank my advisor Prof. Hartmut König for giving me a chance to graduate at Brandenburg University of Technology Cottbus. Without his great patience, manifold guidance, and helpful support, this thesis would have been impossible to complete.

Furthermore, my special thanks go to the other members of the computer network group for their valuable discussions and numerous supports. I thank Michael Meier, Mario Zühlke, Daniel Rakel, Ralf Mahlo, Eduard Popovici, Oleksiy Komar, Alek Opitz, Thomas Holz, Sebastian Schmerl, Joachim Paschke, and Katrin Willhöft. My thanks also extend to the following students for their useful works in supporting this thesis: Robert Krautz, Jiemin Lin, and Guang Yao Liao.

Finally I would like to thank my wife for her firm support in the most important period of my life, and thank my parents and brothers in China for their continual encouragement and concern.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The globalization of the markets and the internationalization of the enterprises are irresistible trends. It becomes more and more a characteristic feature of today's production and development that certain works, e.g. product design or software development, are collaboratively accomplished among geographically dispersed teams in an international enterprise. Communication and collaborative tools are essential means to support this process. Conventional communication means like the telephone, traditional mail, and fax etc. do not meet the multimedia communication demands of the modern business society for their single media communication functionality.

Video conferencing has been always considered as one of the most valuable applications of modern collaborative tools. In a video conference people can speak to each other, see each other, and exchange text, pictures, drawings, and share data in parallel and in real-time, just like having a face-to-face meeting. A survey [1] conducted by RoperASW and TANDBERG demonstrates that video conferencing has a variety of advantages over other communication utilities. It is easier to understand, is more personal, enables quick decisions, builds high trust, makes negotiating easier, reduces confusion and misunderstanding, makes people more accountable, and is better for detailed explanations.

Nowadays many enterprises still spend vast amounts of money on business travels for face-to-face in-person meetings. As an effective collaborative tool having the ability to hold a face-to-face virtual meeting, video conferencing can significantly reduce the need for business travel within an organization or enterprise. As analyzed in [2] video conferencing used as a travel alternative offers benefits not only in the aspect of cost savings but also in the aspect of time savings and quality of life. Video conferencing allows people to instantly communicate with each other anywhere and anytime so that it saves the time invested on a travel meeting, which typically includes flying time, driving time and meeting preparation time. Time savings of the employees are critical for the success of an enterprise since they

can undoubtedly bring the improved productivity and stronger competitiveness to the enterprise. Moreover, as revealed in [2], more than 73% of business travelers find general business travel to be a source of stress. It negatively impacts their life, their sleep, their well-being, and their general performance both before and after their journey. By using video conferencing to replace business travel, an employee's quality of life can be obviously improved, since the stress caused by business travel can be completely eliminated.

In view of possible terrorism and global viral outbreaks, video conferencing has become an important facility to eliminate the risk of business travel. As stated in [3], the trend for the use of video conferences has been rising since September 11, 2001. There was a sharp rise in use after the terrorist attack. The outbreak of SARS (Severe Acute Respiratory Syndrome) in China in March 2003 proved the value of video conferencing again. Video conferencing has become one of the best choices to continue business for companies during difficult times. Due to its face-to-face communication capability people can stay in touch without touching to avoid the SARS infection. As a result, a 125% increase in use of video conferences has been observed compared with average use for 2002 in the report [3].

The deployment of video conferences is not limited to the business area. It can also play a vital role in healthcare, education, and other areas. Doctors scattered around the world could communicate with each other via a video conference, discussing about the medication for a patient in time. Video conferencing also opens up great opportunities for students to select their courses held at other universities. This can break the conventional boundary of the campus to some extent so that it affords students more freedoms and more choices to learn.

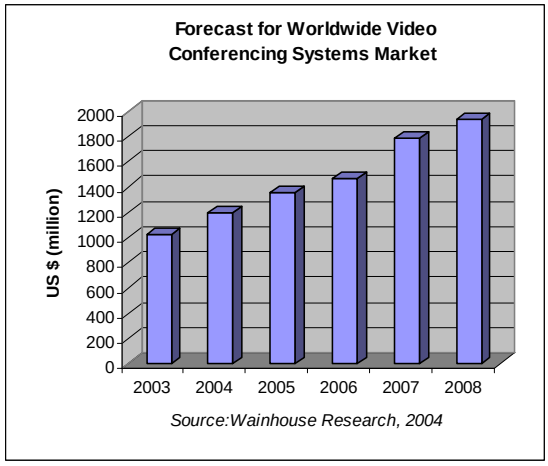


Figure 1.1: Forecast for video conference systems market worldwide

Nowadays the numerous benefits of video conferencing and its significant values have been better understood than before. More and more people utilize video conferencing in their daily working environments. The market of video conferencing systems is in turn expected to show significant growth over the next few years. As predicted in [4] and [5], the world-wide market for video conference systems will grow from approximately 1 Billion US$ in 2003 to more than 1.9 Billion US$ in 2008, producing an annual growth rate of about 8.3% (see Figure 1.1).

1.2 Taxonomy of video conference systems

Video conference systems can be classified in terms of communication technology or equipment form. According to the equipment form, video conference systems can be classified into two categories: room-based systems and desktop systems. From the communication technology perspective, video conference systems can be distinguished between ISDN (*Integrated Services Digital Network*)-based systems and IP (*Internet Protocol*)-based systems[1]. A proposed taxonomy of video conference systems is given in Figure 1.2.

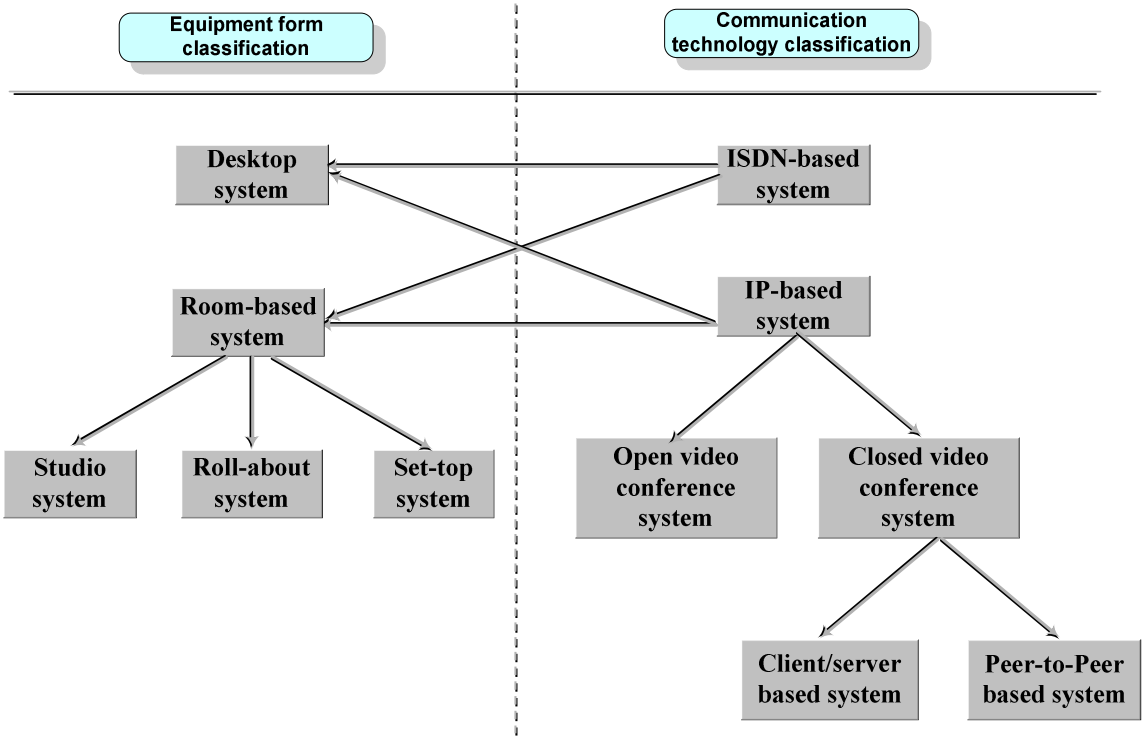


Figure 1.2: Taxonomy of video conferencing systems

[1] In fact, there exists the third kind of system: ATM-based video conference systems. Since ATM networks are rarely deployed in practice, ATM-based video conference systems are being gradually lost their importance in this field.

1) Room-based and desktop systems

- **Room based system:** It works on a custom designed hardware. Thus it presents relatively low ratio of service/cost, since the cost of a room based system is usually high and further it can do nothing except the video conferencing. In view of its mobility, it can further be classified into:

  - *Studio systems* are built into a dedicated conferencing room in a permanent or semi-permanent manner which contains all necessary equipments for running a video conference. It is difficult to set up a spontaneous conference, since the equipment is installed in a dedicated room.

  - *Roll-about systems* are housed in a wheeled cabinet on which all necessary components for video conferencing are placed such as control unit, monitor, camera, and audio system. Such system can be moved from room to room.

  - *Set-top systems*: All necessary components for video conferencing except the monitor and the audio system are integrated into a small compact box. It is designed to be placed on top of a television (TV) to utilize the picture and sound of the TV. A set-top system provides more mobility than the other both systems.

- **Desktop system:** It operates on a general-purposed personal computer (PC) equipped with a cost effective camera, microphone, and loudspeaker or headset. The specific software used for video conferences is installed in the PC in advance. To reduce the computation load on the processor, some desktop systems place the computation-intensive video and audio processing on a Peripheral Component Interconnect card (PCI). The advantage of desktop systems lies in its relatively high service/cost ratio. Basically desktop systems are much cheaper than room-based systems. Moreover, they are applicable not only to video conference applications but also to other applications such as e-mail, web browsing.

The key points of a video conference system are the communication technologies and control mechanisms used. The equipment form classification shows us what the video conference systems look like from the outside. But to know how these systems work inside, a classification from the communication technology perspective is desired.

2) ISDN-based and IP-based system

ISDN-based video conferencing systems run over an ISDN-based network infrastructure. Such systems have been standardized in ITU-T H.320 [6]. ISDN works on a circuit-switched basis. An appropriate number of dedicated ISDN channels need to be set up for a video conference between two end points, since one ISDN B-channel only provides 64 kbit/s bandwidth. Most existing video conference systems are ISDN-based. However, more and more newly emerged systems tend to exploit IP technology for video conference communication. This is mainly for the following reasons [7]:

- *Lower usage fee*: Users of IP networks do not have to pay per minute usage fees. Particularly, video calls within a corporate IP network raise no charge.
- *Converged network*: A major advantage of deploying IP based video conferences is the ability to leverage the primary data network for video conferences resulting in cost savings and increased efficiency.
- *Better availability:* IP networks are ubiquitous nowadays, while this is not true for ISDN networks. Many telephone service providers do not offer ISDN service in every service node.

As shown in Figure 1.2, IP-based systems can be further divided into two categories: open and closed video conference systems [8]. An open video conferencing indicates in this context that a sender in the conference does not know exactly who receives its data. Everyone who subscribes to the multicast address of the conference can join it (unnoticed by the other participants). Open systems usually use the receiver-initiated sending paradigm of IP multicast. Due to its open character of group communication such system is merely suitable to public activities such as open lectures, transmission of conference talks, the delivery of video news, or the distribution of events like concerts or shuttle starts. MBone video conference tools VAT [9], VIC [10] and USMInt [11] are the most popular examples of open conference tools.

Most day-life business activities such as collaborative meetings in companies, medical consultations, court hearings have a confidential character. Closed video conference systems should be applied to these scenarios. A closed video conference means that every sender always knows who receives its data and that only current group members are allowed to send messages to the group. There are two principal approaches for closed video conference systems: the client/server based and the peer-to-peer based approach. The client/server based approach has been well studied and specified by the H.323 standard [12]. H.323 based systems leverage two centralized servers to support closed group meetings: the gatekeeper for the group management and the multipoint control unit (MCU) for the distribution of the media streams. Many commercial video conference systems are in compliance

with the H.323 standard such as Polycom Via Video [13], FVC Click to Meet [14], Vcon MC and Falcon [15], and Microsoft Netmeeting [16]. Although these systems are widely available their drawbacks should not be overlooked. The servers play a key role in the entire system. If they fail, a running conference is terminated. New conferences cannot be held. They represent a so-called *single point of failure*. Servers can also easily become a performance bottleneck, since all control and media messages have to pass through them. They are also attractive targets for attacks. Moreover, they are still pretty expensive what limits their wide deployment.

Peer-to-peer (P2P) solutions present an alternative approach to closed video conference systems. A P2P conference system is characterized by a distributed approach. All group management and media distribution functions of the system are assigned to the peers. The communication runs directly between the peers without passing a server. Thus, the drawbacks in H.323 based systems are basically eliminated. P2P conference systems are well suited to setting up spontaneous conferences, since they put no or little reliance on a certain server. Moreover, P2P conference systems might become the mainstream video conference applications for their fairly low ownership cost. A number of P2P applications with respect to file sharing, distributed computing, collaborative workspace have been developed, but so far only a few P2P video conference systems are reported such as BRAVIS [17], DAVIKO [18], and the P2P-SIP architecture [40].

1.3 Problem statement

The benefit of video conferences to speeding up business activities is widely recognized. Many research efforts have been devoted to designing video conference systems. Numerous related products are available on the market. However, video conference services are not that broadly deployed in the daily workflows like services such as e-mail or WWW. A lot of factors give rise to this result, like high cost of deployment, complexity of the technology, poor understanding of operational requirements, obstinate user habits, lack of necessary functionalities and others. Security and quality of service (QoS) are two imperative functionalities for a successful video conference system. The security of video conferencing is the first concern of users, especially for business conferences which often require confidentiality. If security demands cannot be guaranteed, the users will not use the service. The savings of travel costs with video conference though cannot compensate the potential losses caused by disclosing information to competitors during a video conference. The other issue is the quality of service. A good video conference system should have a sufficient video and audio quality to provide an almost equivalent visual communication effect compared with a face-to-face in-person meeting. Otherwise, users would prefer a travel to an in-person meeting.

In recent years, security and QoS issues in IP network have been intensively studied. This is mainly because the primary design of the IP protocol lacked the considerations of how to protect traffic and how to transfer IP packets with a guaranteed QoS. Following this original design IP packets are delivered on a best effort basis over the Internet. To cope with these issues the Internet research community has developed a series of protocols or communication models to meet practical requirements such as IPsec [19] and SSL [20] for security, and IntServ [21] and DiffServ [22] for QoS. This thesis mainly focuses on security issues of P2P video conference systems. QoS issues of a P2P video conference system will be addressed in another dissertation of the computer network and communication systems group at BTU Cottbus.

It is well-known that the design and implementation of a secure communication system is a nontrivial task. Although ITU-T has ratified the standard H.235 [23] which specifies a security framework to protect H.323 video conferences, only few vendors have announced to support it. Moreover, this framework is not applicable to P2P conferences since the conference servers (MCU and gatekeeper) are involved in accomplishing some security functions, such as authentication and key management. P2P technology is still in its infancy stage. How to protect a P2P video conference over the Internet is still an open issue.

Cryptographic protocols or algorithms are the fundamental elements for building a secure communication system. At least the following two points should be taken into accounts when they are applied to securing real-time video communication:

- **Security:** The ultimate objective of cryptographic protocols or algorithms is to ensure that the traffic securely traverses across the Internet. They should be robust against possible attacks in the Internet, since any security flaws in the design could lead to the compromise of the whole system.

- **Efficiency:** Security always causes additional processing burdens to the system. These burdens may pose a negative impact on the quality of service. For example, a secure conference incurs longer end-to-end communication delays, because messages have to be encrypted and decrypted. Therefore, the deployed algorithms and protocols should be efficient enough to meet the strict QoS requirements of real-time communication.

Additionally a well-designed secure communication system should take the following two aspects into consideration:

- **Easy to use**: This is a critical factor for the success of a system. A system which is difficult to use will be eventually given up by users, even if it is secure and efficient. An ideal easy to use system should assume that users have no professional knowledge about system security requirements. All actions of the users should be only just clicking the mouse.

- **Low costs:** The cost of the system is another important factor which users usually concerns. It relates here not only the ownership but also the usage cost of the system such as administrative cost, bandwidth cost etc.

P2P video conferences are a new emerging distributed application in the Internet. A closed P2P video conference is usually held within a limited scope such as an enterprise or a campus. To assure confidentiality and integrity of such meetings, a secure group communication technology is required. Virtual private networks (VPNs) are such kind of technology. A VPN is a corporate network over public network which can protect applications running over it. Several standard VPN technologies have been widely applied in practice, e.g. IPsec VPN and SSL VPN. The straightforward solution would be that an existing security infrastructure VPN is utilized to support a secure P2P video conference. However, as it will be shown in this thesis, such a solution is not optimal regarding the above stated demands, i.e. security, efficiency, ease of use, and low cost. For that reason, security architecture for P2P video conferencing system is highly desired.

Key management is a keystone in any secure communication systems, since the security of all modern cryptographic protocols or algorithms only rely on the privacy of the used key. All group management tasks including group key management are moved to peers in a P2P system and no dedicated server is placed to be responsible for these tasks. There already exist practical solutions for centralized key exchange protocols relying on a central group key server to deal with the group key management issue, while the development of an efficient and secure decentralized key management protocol in which members themselves manage the group key is still an ongoing research topic.

Another apparent obstacle to realize a secure P2P video conference system is how to encrypt the video stream in real-time. Nowadays, advanced computers are fast enough to encrypt a single channel MPEG-2 video stream with a bit rate between 4 and 9 Mbps in real-time using the naive algorithm approach [24]. However, this evolution of the computer power does not completely eliminate the need to develop faster encryption algorithms for video data. Multiparty P2P video conferences always require specific algorithms for the video encryption because they usually support multi-channel video communication. The simultaneous encryption or decryption, respectively, of all streams causes a huge process-

ing burden at the end systems. Appropriate encryption algorithms allow to alleviate these burdens and to enroll more users to the service

1.4 Goals and organization of the thesis

At BTU Cottbus a P2P multiparty video conference system, called BRAVIS [17], has been developed. It supports closed group meetings and can be used in business talks, project meetings, consultations, teleseminars etc. The closed character of this system is achieved by the following two measures: the entrance into the meeting is by invitation and only current group members are allowed to send messages to the group. Both measures ensure that only desired participators appear in a meeting and conference information is exchanged only within the group. BRAVIS offers the communication privacy comparable to the telephone network. Such a security level can adequately meet the security requirements of some applications, which are inherently not security-sensitive, e.g. teleseminars. But such a security level is far insufficient for a business meeting in which business secrets are usually shared. This is because any information traversing across the Internet is possible to be intercepted in their transmission paths due to its openness. Therefore the appropriate security functions should be integrated into the BRAVIS system to fulfill the strict security requirements of business meetings. To the best of my knowledge at writing this thesis there has been no secure P2P video conference system released worldwide up to now. The basic goal of the thesis is to address this issue by developing a security architecture for the BRAVIS system to ensure confidential talks over the Internet. The other important issues surrounding this basic goal also will be dealt with in this thesis, such as group key management and real-time video encryption.

To sum up, the goals of this thesis are listed as follows:

- **Investigation of the existing VPN technologies**
  It is indispensable to thoroughly investigate the existing VPN technologies which could support secure group communications. Such an investigation should clarify whether or not we can directly make use of the existing VPN infrastructure for securing a P2P video conference.

- **Designing a security architecture for BRAVIS**
  Through the investigation of existing VPN technologies we may recognize that existing approaches are not optimal for use in a P2P video conference. Designing a specific secure architecture is therefore a need for BRAVIS system.

- **Designing a novel group key management protocol**
  The group key management protocols which might be used in a P2P video conference system should be comprehensively examined. This study could indicate the weaknesses of these protocols. To overcome the shortages of these existing protocols, a novel group key management protocol has been designed.

- **Designing a novel video encryption algorithm**
  The specific video encryption methods should be comprehensively surveyed to evaluate whether they are fast enough to keep up with the video stream rates and convenient enough to be integrated into a P2P video conference system. This survey will show that the available methods are not as efficient as expected. To achieve more encryption speed, a novel video encryption algorithm should be developed.

The remainder of this dissertation is organized as follows:

Chapter 2 gives an overview about client/server and peer-to-peer communication models and discusses their differences. Their application in video conference systems is highlighted respectively.

Chapter 3 overviews common attacks in the Internet. The security services which are used to counter against these attacks are briefly introduced. In particular, some possible attacks on a video conference system are described.

Chapter 4 provides insights into all kinds of VPNs which might be used for secure group communications. The necessity to develop the specific security architecture for P2P video conference system is discussed.

Chapter 5 presents a state-of-art security architecture for the BRAVIS system. The functionalities of each security model in the architecture are described.

Chapter 6 systematically evaluates the related group key management protocols in terms of security and efficiency. A novel group key management protocol is proposed.

Chapter 7 surveys the related works about the specific video encryption algorithms. A secure and efficient video encryption algorithm called *Puzzle* is specified.

Chapter 8 concludes this dissertation by presenting the main conclusions and contributions of this work and identifying some issues for further research.

Chapter 2

Client/Server versus Peer-to-Peer

Due to its global connectivity and low-cost usage the Internet has become a popular communication infrastructure. More and more applications are running on top of it. “All over IP” is no longer a hypothesis of the network technology. It has become reality nowadays. Applications over the Internet can be set up either by using the client/server or the peer-to-peer paradigm. The client/server model has been dominant in the Internet application area up to now. The peer-to-peer model is a new approach which emerged several years ago and is increasingly accepted by many applications including video conferencing. In this chapter we give an overview of both models and discuss their differences. We especially consider their application in the context of video conference systems.

2.1 Client/server model

The client/server model is a computational architecture in which nodes are classified either in clients or in servers considering their different functions. Figure 2.1 illustrates the client/server model.

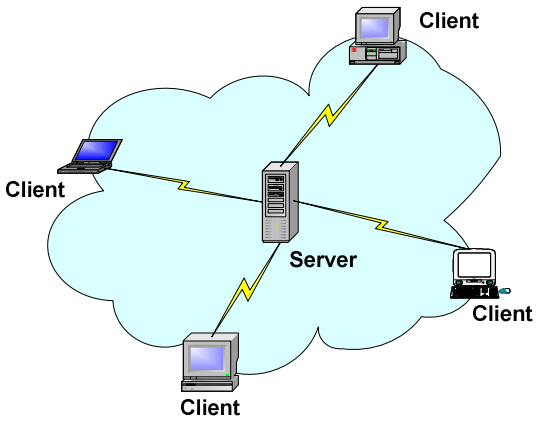


Figure 2.1: Client/Server model

A server offers services to clients. It passively waits for requests issued by clients and returns the results to them after processing these requests. It is, therefore, also called the service provider in the model. Moreover, it can be used to coordinate clients for accomplishing a collaborative task. A client consumes services as follow: it actively sends a request to a server and waits for a response from the server. It is regarded as the service consumer in the model.

The client/server model exhibits the following characteristics:

- **Centralized**: Only the server is the entity to provide services or resources. Clients acquire all information from the server. The direct communication among clients never emerges. The information exchange between clients has to be relayed by mediating servers.
- **Asymmetric**: Clients rely on servers for resources such as files, devices, and even processing power. To be able to process a vast amount of requests from clients, servers basically are equipped with sufficient resources and powerful CPUs (Central Processing Units). This makes servers usually much more expensive than clients.

The client/server model began gaining acceptance in the late 1980s. The commercialization of the Internet in the early 1990s made it more popular because broadly used Internet services such as World Wide Web (WWW) and E-mail have adopted this model.

Although the client/server model has been successfully applied to many Internet applications, it has several remarkable shortcomings. Its centralized architecture renders it prone to failures. All tasks are centrally performed on the server which represents a single point of failure and may easily become a performance bottleneck. The administration and maintenance of the server demands large efforts. Moreover, the asymmetric character of the model leaves the resources of the clients unused. Many clients are often idle most of time.

2.2 P2P model

2.2.1 P2P basics

As an alternative to the client/server model, P2P computing has become a popular term in the IT (Information Technology) area nowadays. The key point of P2P is that the nodes can directly communicate with each other without relying on any intermediate node. Strictly speaking, this concept is not a new idea. The early Internet, the Advanced Research Projects Agency Network (ARPANET) took advantage of this concept to connect four U.S.

universities in 1969. Usenet (1979) is another example in which nodes were directly interconnected without relaying nodes.

The success of Napster [28] made this technology reborn at the beginning of this century. Napster is an online MP3 file sharing application which is used to help individuals to exchange MP3 files over the Internet. It has attracted more than 20 million users in just over a year after its launch. This success sufficiently demonstrates the power and value of the P2P technology.

In the P2P model nodes are usually called peers. They mainly perform symmetric roles. A peer may act both as client and server, i.e., it is not only a service consumer but also a service provider. P2P systems are in general deployable in an ad-hoc fashion rather than in a hierarchical fashion without requiring a centralized management or control [29], [30].

The P2P computing model is characterized by following properties [31]:

- **Decentralized**: The major feature of P2P systems is decentralization. Ownership, control of resources, and administration in a P2P network are distributed among peers.
- **Symmetric**: Peers are given equivalent capabilities and responsibilities. Unlike the asymmetric client/server structure, a P2P system does not require a central coordination among peers so that they can directly interact with each other.
- **Self-organization:** Each peer itself decides which resources can be shared with other peers. Moreover, peers collaboratively establish a transient P2P network according to their interests and needs without central control.
- **Dynamic:** P2P networks are ad hoc in nature. This is based on the fact that peers can join and leave a network at any time.

The most important advantage of the P2P model over the client/server model is that it does not depend on a certain server infrastructure. The decentralized nature of P2P systems can largely avoid or alleviate the problems of client/server systems, such as a single point of failure and the performance bottleneck. In other words, P2P systems are more robust to faults and more scalable than client/server systems. Moreover, P2P systems are more cost-effective due to the elimination of cost-expensive servers.

P2P computing has received a wide attention. It has been increasingly applied in the many application areas such as distributed computing, file sharing and collaborative applications [30]. One of the most cited examples of distributed computing is SETI@home project [32] which collects the idle CPU cycles in the Internet to analyze radio-telescope data for

searching extraterrestrial intelligence. The previous mentioned application Napster [28] is one of successful P2P file sharing applications. AOL Instant Messager [33] and Groove [34] are the representatives of P2P collaborative applications which allow users communicate directly in real time without relying on a server.

2.2.2 Pure and hybrid P2P models

The P2P models are distinguished in pure and hybrid P2P models [30].

Pure P2P models do not rely on any central entities to locate the resources of the network. This task is usually accomplished by using appropriate routing algorithms. Peers in a pure P2P system can be organized using an unstructured or structured topology.

- *Unstructured P2P network:* Peers form a random graph and a given file can be stored at any node. The query is executed through time-to-live (TTL) controlled-flooding routing algorithms. Unstructured P2P networks do not scale well because the complexity of the query is linearly proportional to the number of active nodes N in the network. Namely, the query requires $O(N)$ hops to complete. Figure 2.2 illustrates an example of the file query mechanism in Gnutella [35] which is a typical unstructured P2P system.

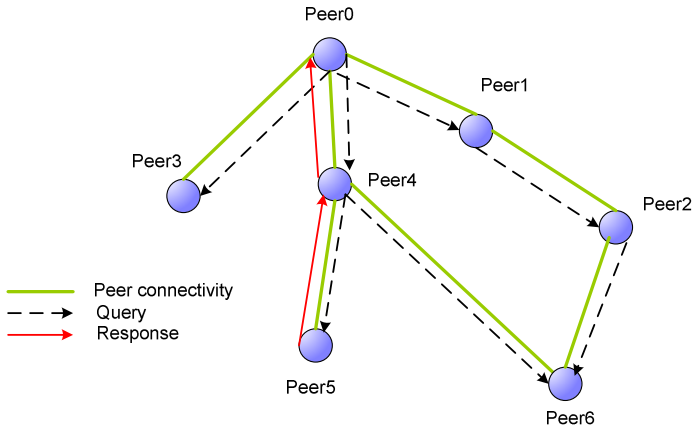


Figure 2.2: An example of the file query in the Gnutella system

- *Structured P2P network*: Peers form a certain deterministic topology which might be a mesh [36], a ring [37], a d-torus [38], and a butterfly [39]. In contrast to the unstructured P2P network, data items (files) are retained not at random peers but at specified peers. This is achieved by using the following measure: each data item is associated with a unique key by hashing its content and stored at the node responsible for that key. The mappings between data items and the associated peers constitute distributed

hash tables (DHTs) which are used like routing tables in traditional network level routing protocols. DHT based routing protocols such as CAN [38], Chord [37], Pastry [177] possess better scalability and query efficiency than flooding routing algorithms used in an unstructured P2P network. The complexity of finding a match query in a structured P2P network is not more than *O*(*logN*). Figure 2.3 shows an example of the routing mechanism in Chord (3-bit identifier).

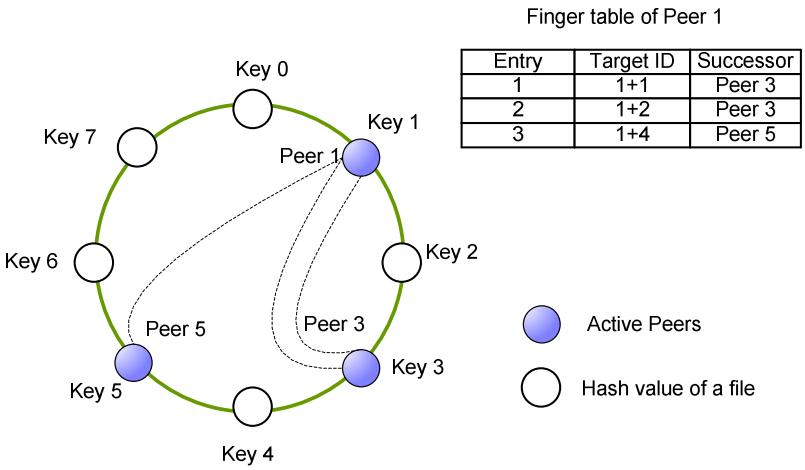


Figure 2.3: An example of routing mechanism in Chord

In Chord each peer and file is assigned an *m*-bit identifier using a common hash function. A peer's identifier is obtained by hashing its IP address, while a file' identifier is generated by hashing the file, which is also called the key for the file. Both kinds of identifiers are ordered in an identifier circle modulo 2^m. A file is stored at the first peer whose identifier is equal to or larger than its key in the identifier space (clockwise organized). For example, given a 3-bit identifier (see Figure 2.3), we assume there are three active peers (peer 1, peer 3, and peer 5) in the network. Peer 1 is responsible for files whose keys range from 6-7, as well as 0-1. Peer 3 deposits files with keys from 2-3 and peer 5 keeps files with keys from 4-5, respectively.

Each peer maintains an *m* entries routing table, called the finger table (see Figure 2.3). For peer *n*, its entries in the finger table point to successors (other peers) which are determined by $(n+2^{i-1}) \bmod 2^m$,where $1 \le i \le m$. For example, in Figure 2.3 the third successor of peer 1 is peer 5 ($(1+2^{3-1}) \bmod 8=5$). Using the finger table, each lookup hop shortens the distance to the target by half, since the target IDs in the table increase nearly by the power of two. This results in a lookup complexity of *O*(*log N*).

Hybrid P2P models use a central server to speed up the resource location in a P2P network. The server is usually called the index server because it stores indexes for all files in the network. The indexes specify the locations of the files. To find a file a peer first connects with the index server to obtain the location of the desired file, and then it sets up a direct link with the peer that stores the file. With a lookup complexity of $O(1)$ this model possesses a higher query efficiency than the pure P2P model. Figure 2.4 illustrates a scenario of this model, where five peers build up a peer group with the assistance of the index server.

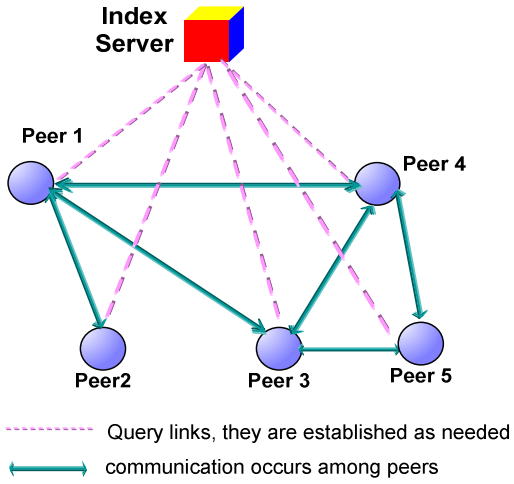


Figure 2.4: Hybrid P2P model

The essential difference between the hybrid and pure P2P model is that the former uses a server for locating the resources, whereas the latter relies on the peers themselves rather than a server to accomplish this task. However, these two models make no difference in the key point of the P2P communication model, i.e. the direct communication between the peers. To some extent, the hybrid P2P model represents also a single point of failure, when the central index server is unavailable. But the negative impact on the hybrid P2P systems caused by this problem is not as serious as in client/server systems in which the whole communication relies on the availability of the server. So the established group communications using a client/server system will terminate once the server crashed. In contrast, a running peer group communication using the hybrid P2P model will not be disrupted even the index server fails. This is because the function of the index is nothing more than the system bootstrapping, and the server does not take part in the direct communication among peers.

2.3 Comparisons of the models

The following table summarizes the main characteristics of the client/server and P2P model which have been discussed in the previous two sections.

Table 2.1: Comparisons between the client/server and P2P model

Features	Client/server model	P2P model		
		Hybrid P2P model	Pure P2P model	
			Unstructured	Structured
Using a central entity	Yes	Yes	No	No
Direct data exchange	No	Yes	Yes	Yes
Complexity of the data query	$O(1)$	$O(1)$	$O(N)$	$O(\log N)$
A single point of failure	Yes	To some extent	No	No
Performance bottleneck	Yes	To some extent	No	No
Scalability	Low	Medium	Medium	High

2.4 Video conference systems

A video conference system is used to enable two or more participants to interactively communicate by simultaneously applying video, audio, and whiteboard. It is also a type of groupware system used for the visual collaboration [178]. As shown in Figure 2.5, a video conference system is usually made up of two parts: signaling and media exchange [8].

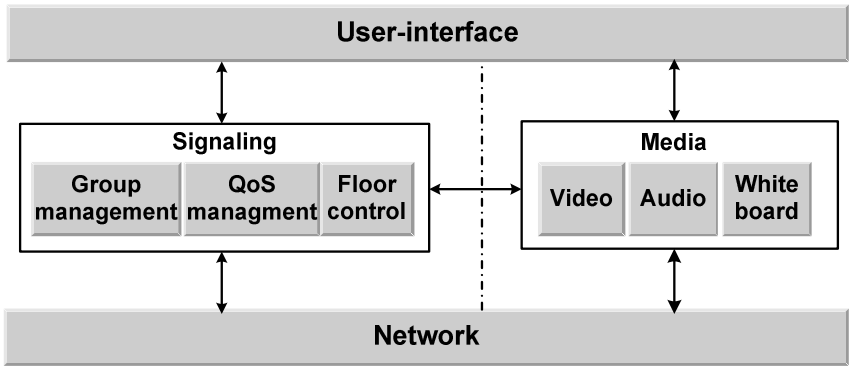


Figure 2.5: General structure of a video conference system

The signaling part is responsible for the conference control. Its typical functions are the group management, the QoS management, and the floor control:

- *Group management*: It is mainly used for the establishment and termination of a meeting. Its functionalities include the supervision of the group composition (joining or leaving a meeting), user registration etc.

- *QoS management*: It negotiates QoS parameters (e.g. the frame rates and the video resolution) between participants with different computing resources and bandwidths, and controls the parameters during the meeting.
- *Floor control*: It regulates the access to shared resources, in particular to the whiteboard, among participants. Basically a shared resource is allowed to be accessed by only one participant at one time to avoid the race condition problem.

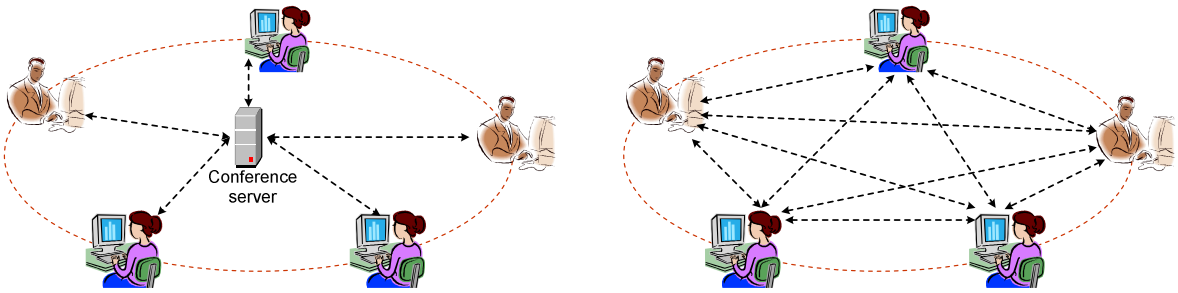
The security management in a further sense also belongs to the signaling part as discussed later in this thesis.

Media

The media part comprises the audio/video processing units and whiteboard communication. The task of the audio/video processing units is to efficiently transmit the audio/video signals over the network. The audio/video processing consists at least of two operations: video/audio digitalization and compression. The whiteboard supports the sharing of documents among participants.

Development of video conference systems

A video conference system can be designed by using either the client/server or the P2P model. In a client/server based video conference system conference servers are introduced to coordinate conference clients by centrally handling signaling data. In some cases even media data are centrally processed by the servers. In a P2P based video conference system no conference server is used to coordinate participants. Instead participants themselves cooperatively perform the conference control by directly exchanging signaling data. Moreover they directly distribute media data to each other. In some cases, a central entity (e.g. register server) is applied to a P2P conference system to make the bootstrapping of a conference easier. Figure 2.6 illustrates the scenarios for a client/server and P2P video conference system.



(a) Client/server based system (b) P2P based system

Figure 2.6: Scenarios for client/server and P2P based systems

A client/server based video conference system suffers the same weaknesses as any other client/server system, in particular that the server represents a single point of failure and may become a performance bottleneck. The peer-to-peer approach offers an alternative way to designing video conference systems. It is a distributed approach, and thereby puts no or only little reliance on a certain server. Thus, the drawbacks of client/server based systems can be basically eliminated.

The difficulty to developing a P2P video conference system is to preserve a consistent view on the meeting at every participant. Only with a consistent view, participants can have a common group membership list, uniquely tuned QoS parameters, and an agreed right to access to shared resources which are essential requirements for holding a conference. This is not a major concern for client/server conference systems, since all signaling is centrally handled by the conference server. The consistent view of a meeting can simply be distributed to the participants by the server. In contrast, a consistent view of the meeting in P2P systems is not trivial to achieve because the processing of the signaling data is distributed to the participants, and no central entities are responsible for this task.

However, most video conference systems have been designed in compliance with the H.323 standard [12] which is based on the client/server architecture. But so far only a few P2P video conference systems are reported like BRAVIS [17], DAVIKO [18] and the P2P-SIP architecture [40], [179]. No document can be found for the DAVIKO system describing how it achieves the consistent view in a meeting [8]. The P2P-SIP architecture allows users to register themselves in a P2P network instead in a central SIP register server. A caller can find a desired communication partner by using a DHT protocol such as Chord. Unfortunately, the proposed scheme is not a complete architecture for a P2P conference system because it just removes the need of an SIP registrar. In addition, there is nothing said about maintaining the consistent view in a meeting. Only the BRAVIS system has addressed the consistent view issue so far which is crucial for the success of a P2P conference system. Therefore, we only give a more detailed introduction to the H.323 and BRAVIS system in the following sections which are typical representatives of the client/server and P2P based systems, respectively.

2.4.1 H.323 video conference systems

H.323 video conference systems were designed by employing the client/server model. The H.323 standard specifies four kinds of logical components (or *H.323 Entities*) used for multimedia communications which consist of one client entity (terminal) and three server entities: gatekeeper, multipoint control unit (MCU), and gateway.

A *terminal* is a client-endpoint (PC or stand-alone device) which provides real-time, bidirectional multimedia communications. The *gatekeeper* is the most important entity of an H.323 network. It also called the administration server which offers call control services and management services to the H.323 entities. The MCU is a central entity which provides services that allow three or more endpoints to take part in a conference call. A *gateway* connects two different networks. It works as a proxy server providing protocol conversion services between H.323 terminals and other terminals that do not support H.323.

In the following we go insight the gatekeeper and MCU because they play decisive roles in a H.323 system. The signaling data for group management, QoS management, and floor control are centrally processed in these two entities. In some application scenarios MCUs are even in the charge of the central processing of media data.

Gatekeeper

H.323 networks are partitioned into different zones for their management (see Figure 2.7). Each zone consists of terminals, gateways, and MCUs. It is managed by a single gatekeeper. A zone may be independent of the network topology and may be comprised of multiple network segments which are connected using routers or other devices. Only one gatekeeper is allowed in a zone at any time.

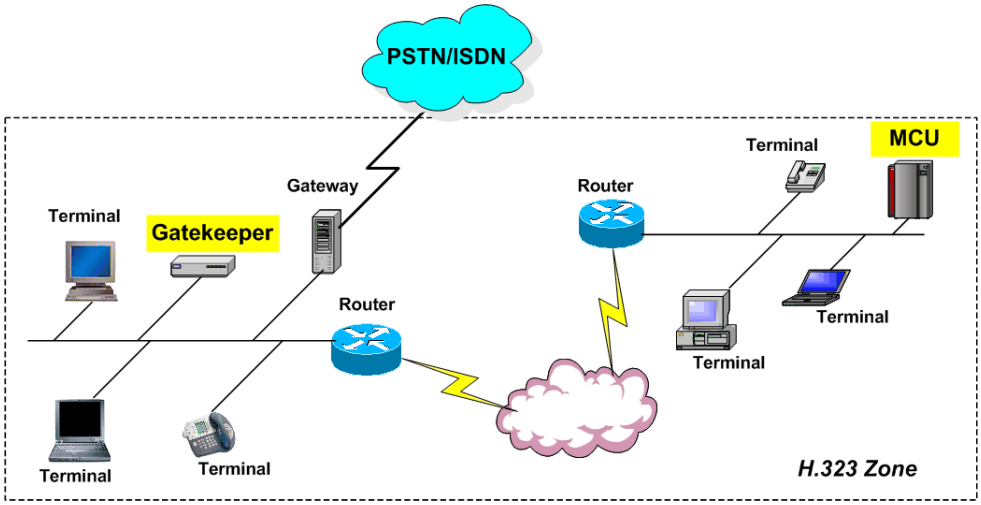


Figure 2.7: A H.323 zone

Although the gatekeeper was originally considered as an optional component in the H.323 standard, it is in practice an essential element for defining and controlling how voice and video communications are managed over IP networks [27]. The gatekeeper acts as an administration server managing all activities in a zone. A gatekeeper supports the group management including user registration, address translation, call accounting, and network man-

agement etc. Functions like call admission control and bandwidth control are used to manage the quality of service in a H.323 zone.

MCUs

A MCU typically consists of the multipoint controller (MC) and the optional multipoint processor (MP). The MC is the conference controller responsible for the consistent view on the meeting. It manages all states of the current conferences, and notifies all participants about participants entering and leaving a conference. The MC also performs the floor control by granting a participant the access right to the shared resources (e.g. to whiteboard) for a certain time. Regarding QoS management the MC determines common capabilities for audio and video processing between all terminals [26]. The H.245 control signaling protocol is the communication channel between the MCU and all terminals to accomplish the tasks mentioned above. The MP provides means for mixing, switching, or other processing of media streams under the control of the MC. MCUs are still expensive. An H.323 MCU may be implemented in hardware or software. Hardware based MCUs are much more expensive. Their costs range from fifteen thousand dollars to over two hundred thousand dollars. Software based MCUs are more moderately priced. The costs are between several thousand and twenty thousand dollars [180].

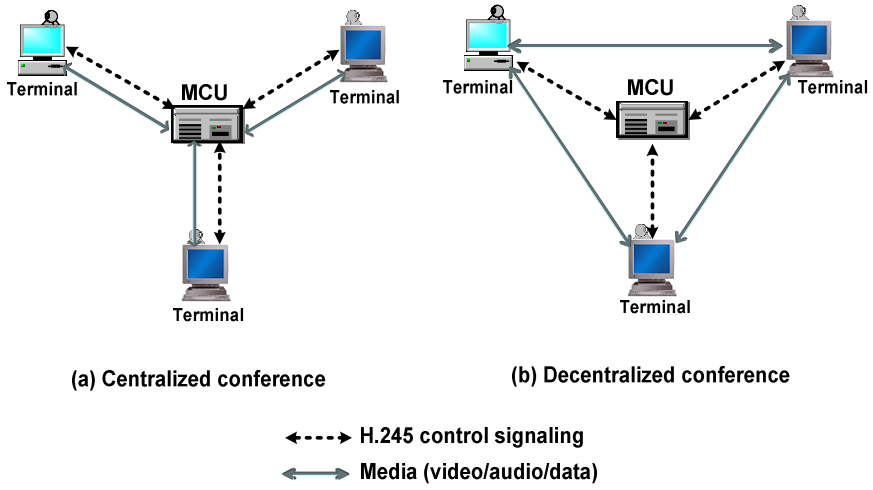


Figure 2.8: H.323 centralized/decentralized conferences

Using MCUs, a multipoint conference can be held in two ways: centralized or decentralized (see Figure 2.8). A *centralized multipoint conference* is one in which all participating terminals communicate in a point-to-point fashion with an MCU. The terminals transmit their control signaling, audio, video, and/or whiteboard data to the MCU. The MC within the MCU centrally manages the conference. The MP within the MCU processes the audio, video, and/or whiteboard data and returns the processed streams to each terminal. In a *de-*

centralized multipoint conference the participating terminals directly multicast or multi-unicast their audio, video, and data to all other participating terminals without passing an MCU. No audio or video MP is required in this case. But similar to a centralized multipoint conference, signaling communications between terminals are still coordinated by a single point MC. The so-called decentralized video conference in H.323 is virtually not a real decentralized conference. Despite its decentralized processing of media data the entire system still falls into the client/server model category due to its centralized processing of signaling data.

2.4.2 BRAVIS system

In fact, early in 1998 when the term P2P was far less popular than nowadays, being conscious of a series of weaknesses of client/server architecture used in video conference systems, at BTU Cottbus a P2P multiparty video conference system, called GSCVA [41], based on ATM was developed. On the basis of GSCVA, an IP version of the P2P multiparty video conference system, called BRAVIS [17], has recently developed. It war designed for supporting small closed collaborative groups up to 20 participants over the Internet in a conference. Small group peer-to-peer meetings are dominant in every-day life such as business talks, conferences, consultations, teleseminars, multiparty games etc. Interactive and collaborative meetings tend to be much smaller compared to the open multicast meetings set up via the Mbone [42]. The term *closed* indicates in this context that peer-to-peer group communication occur in the realm of domains, for instance, inside an enterprise whose branches are geographically dispersed. This corresponds to the so-called federated P2P communication model [43]. The entrance into the meeting is by invitation. This corresponds with every-day procedures which use (oral or written) invitations for participation. The features of the BRAVIS system are the following.

Hybrid P2P model

BRAVIS uses a hybrid P2P model. A SIP registrar is integrated in the system to allow peers registering their current IP address and retrieve the current IP addresses of the other peers for invitations. The hybrid model was chosen for two reasons.

Efficient user lookup

Only one lookup operation is needed to locate a user when using the hybrid P2P model. However using a pure P2P model, this process requires at least $O(logN)$ lookup operations to complete, such as in the Chord system [37]. Moreover, considering the practical situation of a closed environment, this one lookup operation in the hybrid P2P model could be eliminated for most cases if appropriate measures are taken. In enterprises many people are nowadays equipped with desktop computers. So the binding between the user names and IP

addresses will be kept unchanged for a quite long time. People can trivially set up a contact list by caching such binding information during each communication. Therefore, people do not need to connect to the SIP registrar every time if their contact list contains the IP address of the invitee.

Security consideration

In a pure P2P system there is no central authority response for the identity management. This allows an attacker to use different identities to attend conferences. Such an attack is called Sybil attack [44] which means identity forgery. A detailed explanation about this is given in the next chapter.

Decentralized group management

All peers in the group are assigned identical capabilities and properties by using the same system structure. The signaling modules (group management module, floor control module, QoS module), and the media modules (video manager, audio manager, and whiteboard) are available at each peer. Thus, each peer has the ability to supervise the composition of the group, to control the access to shared resources such as the whiteboard, and to tune QoS parameters without calling any additional server. Furthermore, media data transmissions take place among the peers involved in the current conference without using a dedicated server like the MCU in H.323 systems. The signaling modules run on top of GCP protocol [45] [46] (see below). Figure 2.9 shows a four peer conference example and each peer's system structure.

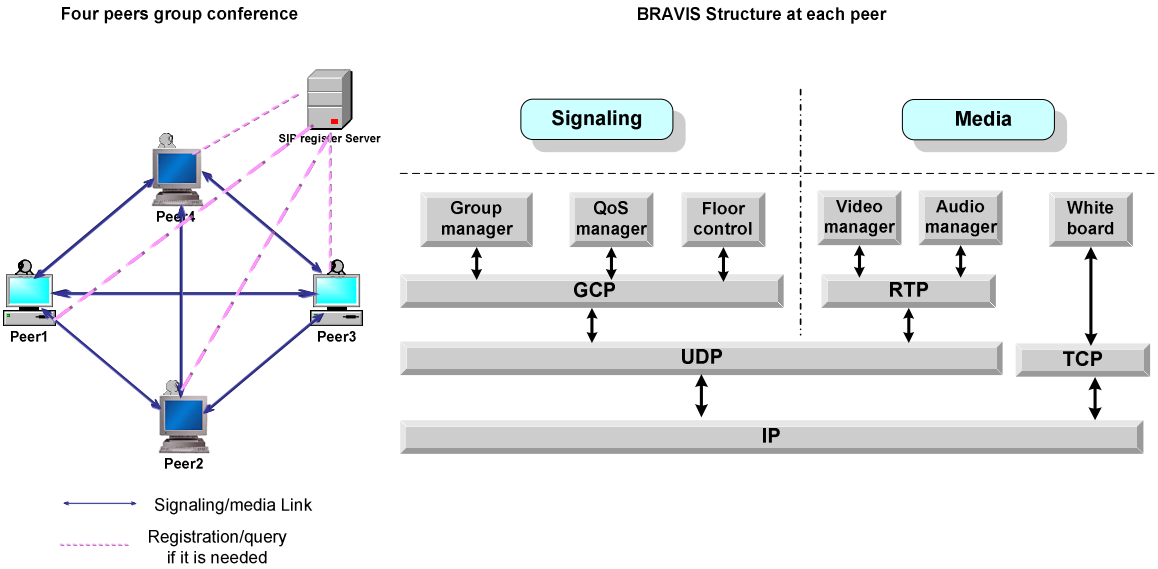


Figure 2.9: BRAVIS system architecture

Decentralized group communication protocol (GCP)

GCP (*Group Communication Protocol*) [45], [46] is the keystone of the entire system. It was designed for addressing a critical issue of P2P video conference systems: how to ensure the consistency between peers. Based on this consistency all peers possess the same view on the actual group state and can uniquely decide all group related issues by themselves, e.g. QoS parameter tuning or floor assignment. For closed groups, the system has further to ensure the closeness of the session. GCP achieves this by providing virtual synchrony [47] service (i.e. reliable, atomic and totally ordered group communication service) to the upper layer modules. Basically, the exchange of signaling data has to fulfill the following requirements to assure the consistency of the group management data:

- **Reliability**: In contrast to the transmission of continuous media data where some frames may be lost, distorted, or discarded, the exchange of control information has to be reliable.
- **Atomicity**: All participants must be equally updated, i.e. the messages have to be delivered either to all participants or to none of them.
- **Ordered delivery**: The messages are delivered in the order they are sent. This ensures that events like joining or leaving of a conference, or assigning the floor are indicated to all participants in the order they occur.

2.4.3 Comparisons between client/server and P2P conference systems

In the following we give some concise comparisons between client/server and P2P video conference systems by using H.323 systems and BRAVIS as examples. The comparisons are related to decentralization degree, fault-tolerance, cost of ownership, and scalability.

➢ *Degree of decentralization:* The client/server based video conference system is a centralized system which relies on central servers to deal with the signaling and media data of all participants during the meeting such as gatekeeper and MCU in the H.323 system. The P2P video conference system is highly decentralized. No dedicated servers are used to centrally process the signaling and media data. This has been demonstrated in the BRAVIS system.

➢ *Fault-tolerance:* The client/server video conference system is not fault-tolerant due to the application of servers. Whenever the servers are out of service, all existing conferences will be terminated and the new conferences cannot be held. Whereas the P2P video conference system does not have this problem, since all control and media data are exchanged directly among peers during the meeting without the intervention of a

server. The failure of a peer only prevents that peer from using the conference service, but does not affect other peers' normal operation.

- *Cost of ownership:* The cost of a client/server video conference system is usually much higher than that of a P2P system. This is because expensive servers have to be applied to coordinate the communication among all participants as e.g. MCUs in H.323 systems. In contrast, P2P conference systems like BRAVIS handle the coordination of all peers in a conference by themselves. Thus no money is required for expensive servers.

- *Scalability:* The number of conferences supported concurrently in a client/server based conference system is limited due to the performance restriction of the servers. Whereas their number is not restricted by such a condition in a P2P system, since no central servers are deployed. The number of simultaneous conferences supported in a P2P system is constrained not by the system itself but by the available bandwidth of the network. Thus from the perspective of the capacity of the system, a P2P video conference system is more scalable than a client/server system.

The above comparisons have shown that P2P based video conference systems possess a number of benefits over client/server based conference systems. So it is likely that many future video conference systems will be designed by applying the P2P approach.

Chapter 3

Threats to Conference Systems and Countermeasures

The Internet has become an important part of our life. It brings both opportunities and challenges. The Internet promotes efficient business practices and facilitates information exchange. On the other hand, security is becoming a major concern for its use due to its open nature. This chapter investigates the possible threats to video conference systems when running over the Internet by using BRAVIS system as an example and discusses respective countermeasures.

3.1 Threats in the Internet

Security has added as an afterthought to the Internet. The ARPANET, the predecessor of the today Internet, connected a small group of people within several US universities. Security was less an issue at that time, since users generally knew and trusted each other. This ideal scenario was no longer preserved after the Internet was commercialized in 1989. Nowadays the Internet has already penetrated into every corner of the world and its usage patterns have radically changed. Its use is not limited to the relatively closed academic community any more; instead it has connected diverse communities around the world. Hence the basis to forming trust relationships among users is lost. The Internet is essentially a hostile network environment nowadays. The report [48] shows that the number of security incidents occurred in the Internet have grown dramatically year by year. Thus it is commonly believed that security is a key point for the correct utilization of the Internet.

The Internet is more strongly vulnerable to attacks than traditional communication networks such as ISDN. The latter is a tightly controlled and accountable network where users are administrated by their service providers. The Internet, on the other hand, is a complex, dynamic world of interconnected networks with no clear boundaries and no central control [49]. Therefore adversaries can mount attacks anywhere without great fear of identification.

Widely available automated attack tools open more opportunities for attackers. Malicious users with little technical knowledge can effectively attack targeted objects with these tools.

Attacks may be passive or active. *Passive attacks* access without authorization to information by silently eavesdropping packets running over the network. Ethereal [50], a packet sniffer tool, can be, for example, used for such attacks. Passive attacks are difficult to detect, because they do not change data. *Active attacks* actively impair the normal operation of a system by interrupting, modifying, or generating network traffic. They are easy to detect but difficult to prevent completely. The kinds of active attacks include but are not limited to:

- **A masquerade attack** aims at impersonating an entity.
- **A replay attack** resends a previous recorded message in its entirety or in part to produce an unpredictable effect.
- **A man-in-the-middle attack** means that an attacker intercepts and intently alters exchanged data with the aim to masquerade as one of the communicating entities to fully accessing to a service or resource.
- **A modification of a message** occurs where the content of a message is randomly and deliberately altered without detection and leads to an unauthorized effect.
- **A denial-of-service attack (DOS)** prevents legitimate users from accessing the usually available resources by either overwhelming the target network with a large amount of data or deliberately consuming a limited resource [51]. The UDP (User datagram protocol) flooding attack is an example of exploiting the former attack technique which simply injects many UDP packets into the network to consume all available network bandwidth. The SYN flooding attack is a typical example of using the latter attack technique where an attacker initiates a lot of TCP (Transmission control protocol) connection requests without completing the required handshakes as required by the protocol. This can exhaust the resource of the victim machine, thereby preventing other users from setting up connections with this machine.
- **A distributed denial-of-service attack (DDOS)** is accomplished by capturing a large number of machines and forcing them to attack a target machine using one certain DOS attack technique. The difference between DDOS and DOS is that the former makes use of a number of compromised machines for the attack.

3.2 Threat modeling for BRAVIS system

Like other services video conference services are prone to attacks appearing in the Internet. The main focus of this thesis is to construct a secure P2P video conference system to support confidential talks over the Internet. In general, the first step towards building a secure application is to identify risks raised by possible attacks. A certain attack could pose the

unequal degree of impairments on different applications, because each application has its own characteristic, scope of deployment, and security requirements. Thus each application or system necessitates an associated risk analysis to specifying which kinds of threats it primarily and truly envisages.

Threat modeling is a structured methodology to perform the risk assessment for an application or system. It systematically identifies and models the threats that an application most likely faces. So far there exist two approaches used for this purpose: attack tree and STRIDE model. The attack tree model (or threat tree model) [52] enumerates all possible threats to a system and summarizes them into a tree structure. The root of the tree corresponds to the intended goal of an attacker. The leaves represent different ways to achieving that goal. Each intermediate node is a sub goal, and children of that node are ways to achieving that sub goal. This model works well for small systems. However, applying this model to complex systems is labor-intensive and nearly infeasible, since there may be thousands kinds of possible attacks faced by a system. Moreover, new kinds of attacks unceasingly emerge with the evolution of the attack techniques. Consequently, the designed attack trees have to be revised due to those new attacks.

Recently Swiderski and Snyder proposed the STRIDE model [53] for assessing the damages caused by threats. The STRIDE model does not exhaustively list all possible attacks, but focuses on the results of possible attacks to a system. It categorizes all possible attacks into six classes based on their effects: **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, and **E**levation of privilege. This model has been applied to many applications for risk analyses [54], [55], [56]. Likewise we make use of it for the threat assessment to our BRAVIS system.

The BRAVIS system may envisage five kinds of the threats of the STRIDE model (see Figure 3.1). Repudiation is not a relevant threat for BRAVIS. It means that a user denies that he/she performed specific actions or transactions. This threat to a video conference system implies that a user refuses to pay for a video conference service by disallowing that he/she has utilized it when the system is deployed for the provision of public services. The appropriate countermeasures to defending against this threat have to be taken by a video conference service provider, because he/she will earn money. Such a threat can be neglected for BRAVIS, since it is intended to be used in an enterprise environment where video conference services usually raise no charge.

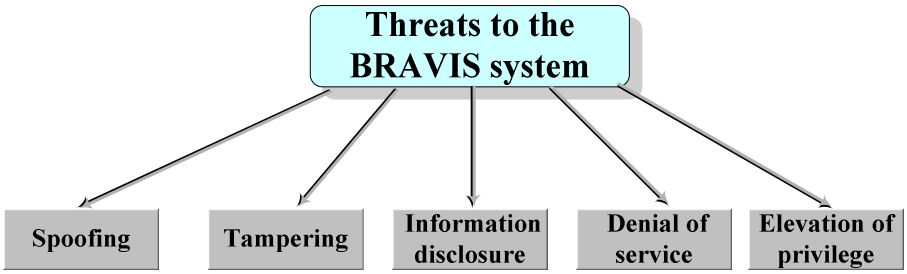


Figure 3.1: Threats to the BRAVIS system

These five threats may affect the BRAVIS system as follows:

- **Spoofing.** An attacker may participate in a video conference using someone else's identity. This can be accomplished by using a masquerade or a man-in-the-middle attack.
- **Tampering**. Signaling or media data of a video conference may be changed in transit by an attacker. This can terminate the conference or degrade the quality of the conference. For instance, a serious corruption of voice data might reduce the clear understanding of the talk by participants.
- **Information disclosure.** The content of a meeting might be intercepted by an attacker. This can be readily done by means of a packet sniffer.
- **Denial of service**. A legitimate participant sometimes might be unable to initiate a video conference call or answer to it due to a DOS or DDOS attack.
- **Elevation of privilege.** A participant attempts to achieve a higher level of privilege than he/she is normally allowed to have. For example, in an enterprise setting, employees are usually not allowed to place a video conference call bypassing the immediate superiors.

3.3 Countermeasures

3.3.1 Security services for the BRAVIS system

To mitigate the threats mentioned above, the following security services ought to be included in the BRAVIS system to ensuring the security of the conferences:

- **Authentication**: Each participant can enter a meeting only after his/her claimed identity is verified. This ensures the participants in a meeting are the people they claim to be.

- **Integrity:** All messages exchanged in the meeting should not be changed during transmission. This concerns both: signaling and media data messages.
- **Confidentiality:** The content of a meeting should only be accessible by current participating members. It should be never disclosed to a non authorized third party.
- **Authorization**: The participants are only allowed to enter a meeting that has granted the right to them.

These four security demands are supposed to withstand the threats spoofing, tampering, information disclosure, and elevation of privilege, respectively. How to eliminate the risks raised by denial of service will not be taken into account in this thesis for two reasons: (1) A DOS attack poses less impact on a P2P based system than a client/server based system. The BRAVIS system follows the hybrid P2P model. A SIP (Session initiation protocol) registrar server was introduced to provide off-line services such as IP address registering. The unavailability of this server due to a DOS attack does not affect the running conferences, because the communication runs directly between peers. (2) No general-purpose measure can be used to actively defend against DOS attacks due to their enormous kinds of variations. In practice precautionary measures are taken to detect them rather than mitigate them. For instance, an intrusion detect system (IDS) could be deployed to automatically detect and respond to DOS attacks. The possible threats and the corresponding countermeasures in the BRAVIS system are summarized in Table 3.1.

Table 3.1: Threats to the BRAVIS system and corresponding countermeasures

Threats	Countermeasures
Spoofing user identity	Authentication service
Tampering with data	Integrity service
Information disclosure	Confidentiality service
Elevation of privilege	Authorization service

3.3.2 Cryptography

Cryptography is a commonly used technology to deliver security services required by a system. It is a mathematical based technology that transforms data into the unintelligible form to prevent undetected modification or unauthorized access [57]. The BRAVIS system provides the above mentioned security services by using cryptography technology. The basic and widely used cryptographic methods embrace symmetric key algorithms, asymmetric key algorithms, hash functions, and digital signatures.

Symmetric key algorithms

Symmetric key algorithms assure confidentiality by transforming a plaintext into an unreadable form. The name of symmetric key algorithms originates from their fundamental property that the same key is used in the encryption and decryption process (see Figure 3.2). Symmetric key algorithms require that the shared secret key is distributed to the communicating parties via a secure channel, because the privacy of the communication mainly depends on the secrecy of the key. Therefore symmetric key algorithms are also called secret key algorithms. Secure key exchange channels for the delivery of keys are usually difficult to set up between people who never met before.

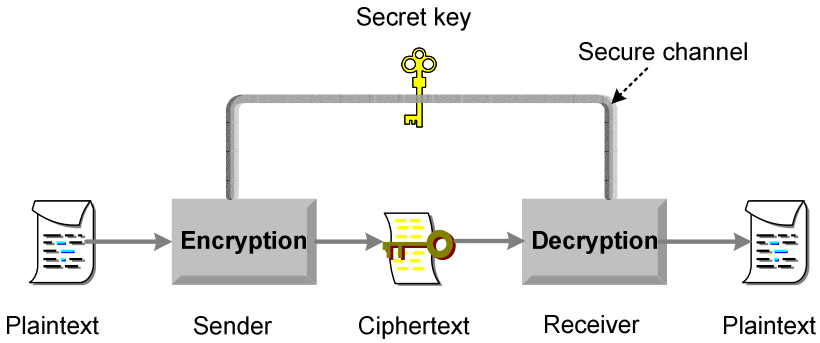


Figure 3.2: Symmetric encryption

Typical symmetric key algorithms, such as DES (Data encryption standard) [58] and AES (Advanced encryption standard) [59], are basically constructed by using substitution operation in conjunction with transposition operations to archive the maximum security. A substitution operation replaces a plaintext symbol by another one. A transposition, also called permutation, rearranges the order of symbols. Substitution and transposition operations serve to achieve confusion and diffusion, respectively, which are two essential attributes required for a secure symmetric key algorithm [60]. Confusion means that any relationship between the plaintext, the ciphertext, and the key are concealed. Diffusion refers to the property that the statistical structure of the plaintext is not retained in the ciphertext.

Asymmetric key algorithms

Asymmetric key algorithms introduce a pair of keys for each participant: a public and a private key. The public key can be publicly and widely disseminated, while the private key must be kept secret by the owner. Therefore these algorithms are also referred as public key cryptographic algorithms. The public key is generated from the private key, but the private key cannot be determined from the public key. A plaintext encrypted using the public key can only be decrypted with the associated private key, and vice versa. The primary benefit of asymmetric key algorithms is that the need for the secure key-exchange channel is elimi-

nated, so that people who have no pre-existing security arrangement can securely exchange messages. Asymmetric key algorithms can be used for confidentiality as well as authentication purposes. Figure 3.3 illustrates their use for confidentiality.

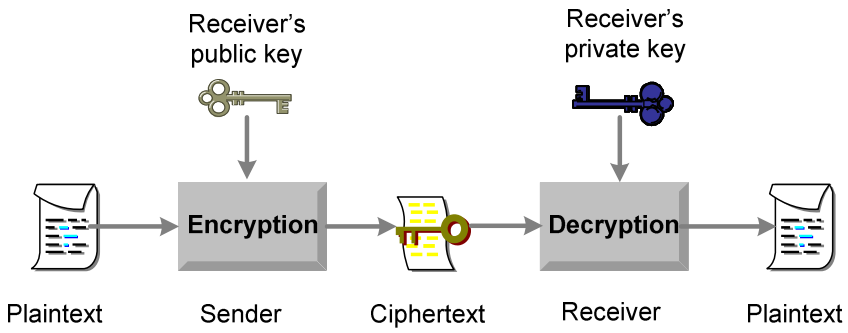


Figure 3.3: Asymmetric encryption

Asymmetric key algorithms are different from symmetric ones in the way they provide security. The security of asymmetric key algorithms completely relies on the difficulty to solve some well-known problems in number theory. The famous RSA (Rivest-Shamir-Adelman) algorithm [61] was established on the basis of the problem of factoring large numbers. The Diffie-Hellman (DH) algorithm [62] which is broadly used in key exchange protocols depends on the difficulty of computing discrete logarithms in a large finite field. Asymmetric key algorithms are usually in the order of 1000 times slower than symmetric ones due to their larger computation complexity. Moreover, asymmetric key algorithms need larger key sizes than symmetric ones to achieve equivalent security strength. The following table gives a comparison of the different key sizes for the same security level for the two kinds of algorithms.

Table 3.2: Key size comparison at the equivalent security strength [63], [64]

Symmetric Algorithms (DES, AES)	**40 bits**	**56bits**	**64bits**	**80bits**	**96bits**	**112bits**	**120bits**	**128bits**
Asymmetric algorithms (RSA, DH)	**274 bits**	**384bits**	**512bits**	**1024bits**	**1536bits**	**2048bits**	**2560bit**	**3072bits**

Hash functions

Hash function is a computationally efficient function that maps an arbitrary length message to a fixed length value. For instance, the widely used SHA-1(Secure Hash Algorithm 1) algorithm hashes a variable-length message to a 160 bit value. A hash function must have the following properties when employed in cryptography:

- **One-way**: Given a hash value, it should be computationally infeasible to derive the corresponding original message.

- **Collision resistance**: It should be computationally infeasible to find two different inputs that hash to a common value.

These properties assure that each message has a unique hash value associated with it. So the hash value is also regarded as "digital fingerprint" or "message digest" of a message. Hash functions can be used for checking the data integrity of a message in a local host. However, they alone are unable to protect the data integrity of a message transmitted over a network. An attacker could insert his own message and the hash value of that message into the network to cheat the receiver, because the receiver has no means to determine whether a message comes from the proper communication partner or not.

In practice keyed hash functions, also called *message authentication codes* (MACs), are used for data integrity checks of networked applications. The MAC of a message is computed by either encrypting the hash value of the message with a joint secret key or hashing the concatenation of the message and the secret key. Thus any alteration of the message including message fabricating can be detected by the recipient, because the MAC value is tightly bundled with the secret key that the attacker does not know. HMAC [65] is a stronger variant of the MAC method. It applies keyed hash functions twice in succession to generate the hash value of a message.

Digital signatures

Message authentication codes protect the message integrity of two communicating parties, but they do not protect the two parties against each other [66]. This is because the two parties share the same secret key, so that each can deny that he/she actually sends the information or forges a message on behalf of the other. To prevent these possible disputes, digital signature schemes have to be introduced which use an asymmetric key algorithm.

Digital signatures are usually created in two steps. First the sender calculates the hash value of the message, then he/she encrypts (signs) the hash value with his private key. Signing the message digest instead of signing the message itself allows to make the digital signature shorter and to reduce the computation delay, since the length of message digest is fixed regardless of the length of the message. Moreover, a hashing function is much faster than a signing operation in practice. To verify the digital signature, the receiver first re-computes the hash value of the message and then decrypts the digital signature with the respective public key of the sender and compares this decrypted hash value with the recomputed one. If the two values match the signature is valid. Figure 3.4 illustrates the process of creation and verification of a signature.

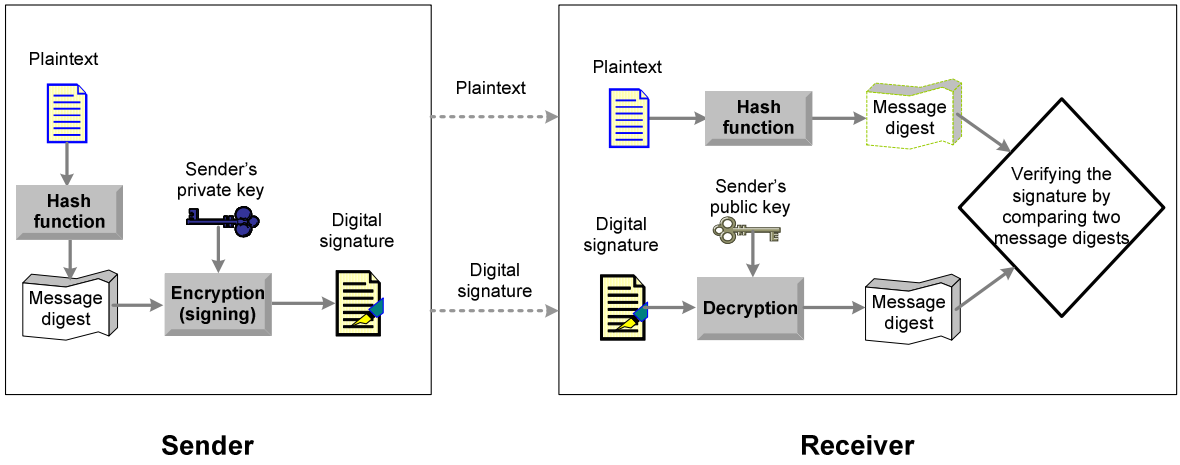


Figure 3.4: Digital signature

Digital signatures are mainly used to verify the authenticity of the origin of information (data origin authentication and non-repudiation), since the sender's private key uniquely identifies the sender. As a side-effect of a digital signature, the data integrity of a message is checked as well, because the signature is verified by comparing the received hash value with the recomputed one. Moreover, digital signatures have become an essential element in a signature-based authentication protocols for entity authentication.

3.3.3 Entity authentication

Entity authentication is to verify the authenticity of the claimed identities of communicating entities. It plays a crucial role in a secure system. Providing other secure services such as data confidentiality, data integrity makes no sense without entity authentication. The privacy of a system would be wholly undermined if an adversary could gain access to it with a forged identity.

Entity authentication verifies the credentials presented by a claimant. Thus entity authentication technologies can be classified into three main categories based on the type of credentials [67]:

- **Something you know**. The claimant presents a common secret to the verifier, e.g. a password or a signature where the possession of the secret is verified.
- **Something you have.** The claimant demonstrates the possession of something to the verifier which is only retained by the claimant. Something typically is a physical object that is difficult to forge or to modify, such as magnetic-stripe card, smart card, and so on.

- **Something you are.** The verifier measures physical and behavioural properties (biometrics) of the claimant, such as fingerprints, handwritten signatures, voice, and so on. These biometric values are unique for every individual in the world.

The "something you have" class of authentication technologies requires the corresponding reader to retrieve the information stored in the card when authenticating. However, readers are generally not embodied into computer systems as standard components. So this kind of technologies is not widely deployed in computer and network environments.

The "something you are" class are not advocated to be used for user authentication via remote authentication servers, because the corruption or compromise of authentication serves gives rise to large-scale privacy and security issues. An attacker can utilize the stolen biometric values of an individual to impersonate him/her. The more serious problem is that it is difficult to stop such impersonation attack even if the compromise of the biometric values is perceived. This is because the biometric values of an individual are everlasting for one life without change. Therefore this kind of technologies is usually used for local authentication, for example, to control access to a laptop.

Actually so far only the "something you know" class is widely employed in network environments for authentication purpose. Password-based authentication and signature-based authentication are two most common used techniques in this class. There is a need to determine which one is more appropriate for peer-to-peer network environments.

Password-based authentication works fine in client/server environments. Its principle is rather simple: passwords are installed in the server in advance; the server verifies the authenticity of the client by comparing the stored password of that client with the received password. To avoid interception and replay of a password over an insecure channel, challenge/response schemes are designed to achieve the goal. In a challenge/response authentication protocol a client can prove the possession of a secret to the server without transmitting it over the network, e.g. Challenge handshake authentication protocol (CHAP) [68] and the Needham/Schroeder authentication protocol [69]. However, a password-based authentication approach inherently faces four issues for its use in P2P network environments due to the lack of an authentication server to centrally manage passwords. First, it does not scale well, since each peer has to store the passwords for other peers. Secondly, its use is confined between known peers, because passwords should be securely distributed in an out-of-hand manner before communication. Thirdly, a malicious peer can impersonate other peers after exchanging passwords with it. Moreover, compromising a peer may disclose all passwords to an adversary, so that all passwords in the system have to be updated. For short, the password-based authentication approach is not a good choice for a P2P system.

The *signature-based authentication protocol* is designed on the basis of a digital signature. The claimant demonstrates the possession of its private key by signing a message with it. The verifier can ascertain the validity of the signature by using the respective public key of the claimed identity. Moreover, to prevent replay attacks, additional nonces or timestamps are attached to the messages exchanged. This measure provides timeliness information about when an authentication message was created so that a peer can determine whether another one has actually participated in the current communication [70]. The signature-based approach completely overcomes the weaknesses shown in a password-based approach when it is applied to a P2P system. The signature-based authentication protocol is a general large-scale solution, because each peer needs only to keep its own private key. The public keys of all peers are published in a public directory. Peers which never met can authenticate each other due to the free accessibility of public keys. A malicious peer cannot impersonate another peer, since each peer keeps its private key securely. The compromise of one peer only threats the private key of that peer so that no private key is needed to update with the new one except that of the compromised peer. To sum up, the signature-based approach is more suitable for a P2P application than the password-based approach when used for entity authentication.

3.3.4 Roadmap of applying crypto algorithms in BRAVIS

In practice most systems simultaneously apply secret key and public key algorithms to provide security services to profit from the complementary properties of both kinds of algorithms. These systems are called hybrid cryptographic systems. Secret key algorithms provide higher performance in term of encryption and decryption operations, but the secret key distribution is a big issue. Public key algorithms are in contrary relatively slow in their use. The primary advantage of public key algorithms is that no secret channel is needed for the key distribution, because the keys can be publicly distributed. A hybrid system is usually built by combining the advantages of both kinds of algorithms. The message encryption is performed by an efficient secret key algorithm. A public key algorithm is employed for the distribution of the secret key needed in the secret key algorithm. Such design strategy has been deployed in lots of applications. For example, PGP (Pretty Good Privacy) [71] is a hybrid system used for secure e-mail. E-mail information itself is encrypted with a randomly generated session key, while this session key is encrypted by using recipient's public key. Finally the encrypted e-mail and the encrypted session key together are forwarded to the recipient.

Following this principle, we design a crypto system for the protection of video conference services offered by the BRAVIS system. It is helpful for us to figure out a roadmap that illustrates the relationship between crypto algorithms applied in the system as well as the

relationship between crypto algorithms and the desired security services. Figure 3.5 depicts the roadmap of crypto algorithms used in the BRAVIS system.

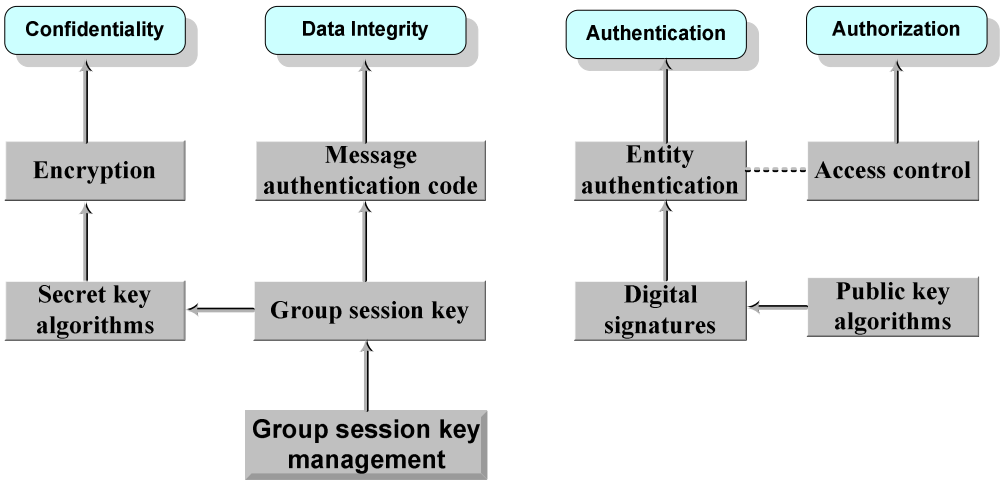


Figure 3.5: Roadmap of crypto algorithms in BRAVIS

As shown in Figure 3.5, confidentiality for all kinds of messages in a video conference is ensured by means of secret key algorithms to meet the strict real-time requirements. Although both MAC algorithms and digital signatures can be used for the data integrity purpose, MAC algorithms are selected due to their more computational efficiency. As discussed in Section 3.3, signature-based authentication rather than password-based authentication schemes should be applied to P2P systems for entity authentication. Signature-based authentication schemes are usually time-consuming, because they are based on computationally-intensive digital signatures. Nevertheless signature-based authentication schemes are acceptable in real-time applications, because the entity authentication is only performed when a new member wants to join a meeting. The slow authentication schemes do not hurt the real-time communication of a running conference except delaying the joining of a new member. Entity authentication and access control list (ACL) are closely linked. The authorization service is achieved by checking ACL to decide whether an authenticated participant is permitted to perform some actions predefined in the ACL.

The privacy of a meeting largely relies on the secrecy of the used group session key as it is an import input to the secret key algorithms and MAC. The compromise of the group session key gives rise to the loss of data confidentiality and integrity in a meeting. Designing an efficient and secure group key management protocol is still a challenging task for P2P real-time settings. For this, we propose a simple and new protocol, called VTKD (*virtual token based key distribution*) in Chapter 6 to address this critical issue. The primary advantage of VTKD is that it is more efficient in terms of key renewal delay than existing

schemes. This results from the fact that not only public key algorithms but also secret key algorithms are involved in the VTKD protocol. Moreover, secret key algorithms are mainly used in the rekeying procedure.

The roadmap of Figure 3.5 has specified the types of crypto algorithms and protocols used to provide the desired security services for the BRAVIS system. In other words, it has enumerated all necessary architectural elements needed for the development of the security architecture of the BRAVIS system. How these elements are efficiently integrated into the system is addressed in Chapter 5.

As mentioned above, public key algorithms can be used in the group key management protocol for the secure delivery of the group session key, but they introduce an equally difficult problem, i.e. how to disseminate the public key. It is true that public keys can be published and distributed freely without via secure channels, but how to ensure that a public key really belongs to its owner is an issue. Delivering a public key over an open communication channel without proving the ownership is dangerous. Man-in-the-middle attacks are the potential threats in which an attacker replaces the transmitted public keys with his public key, so that he can gain access to the communication, while the real communication partners remain unaware of that. Therefore, public keys have to be distributed in a certificate form which binds a public key to identifying information about its owner. This has created a great research area, because the certificate management (generation, distribution, trust management, revocation) is a sophisticated issue. In the next section we give a short overview as far as it is needed in this work, but it is not the focus of this thesis.

3.4 Certificate management

The certificate is a cryptographic data structure used for binding a subject's identity to its public key. The most popular certificates are the PGP [71] and the X.509 certificate [72], [73].The former is merely used in the secure E-mail application PGP, though its format is understood in many other protocols (e.g. IPsec). In contrast, the latter has been strongly recommended and broadly deployed in many applications or systems. Its application scope is not limited to the secure E-mail application (e.g. S/MIME). Many well-known security protocols such as IPsec, SSL make use of it for entity authentication. A X.509 certificate format is shown in Figure 3.6.

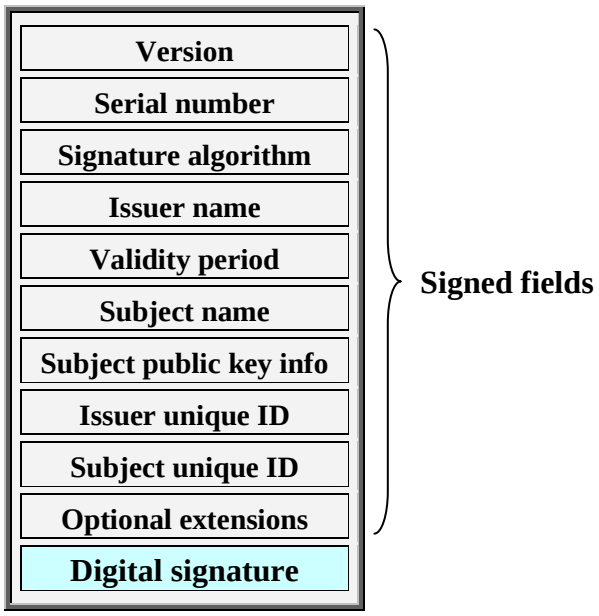


Figure 3.6: X.509 certificate format

All information fields in a X.509 certificate are signed by a trusted third party (TTP) to prove its authenticity. It is usually assumed that the public key of the TTP is widely available. A user can verify the signature of a certificate by using the public key of the TTP so that he/she can confirm the validity of the certificate. The PGP certificate format differs from X.509 certificate format in some information fields, but they are common in the following five fields which are vital for the binding a public key to a subject name (user name): issuer name, validity period, subject name, subject public key, and signature. X.509 certificates are usually managed in a centralized way, while PGP certificates select a distributed manner for their maintenance.

3.4.1 X.509 certificate management

The X.509 certificate management relies on the existence of a public key infrastructure (PKI) [74], which has a centralized architecture. The main purpose of a PKI is to manage public key certificates and to make them widely available for a community of users in an application of asymmetric cryptography. The functions of a PKI primarily include the creation, the revocation, the storage and archival of public key certificates. The main components of a PKI are shown in Figure 3.7.

A PKI consists of the following components:

- **Certificate authority (CA)** is the issuer of certificates and the certificate revocation lists (CRLs).
- **Registration authority (RA)** is an optional component used to undertake some administrative functions from the CA, mainly associated with the subject registration process.
- **Repository** is the directory to store X.509 certificates and CRLs, and to make them publicly available.
- **Subjects** are certificate holders.

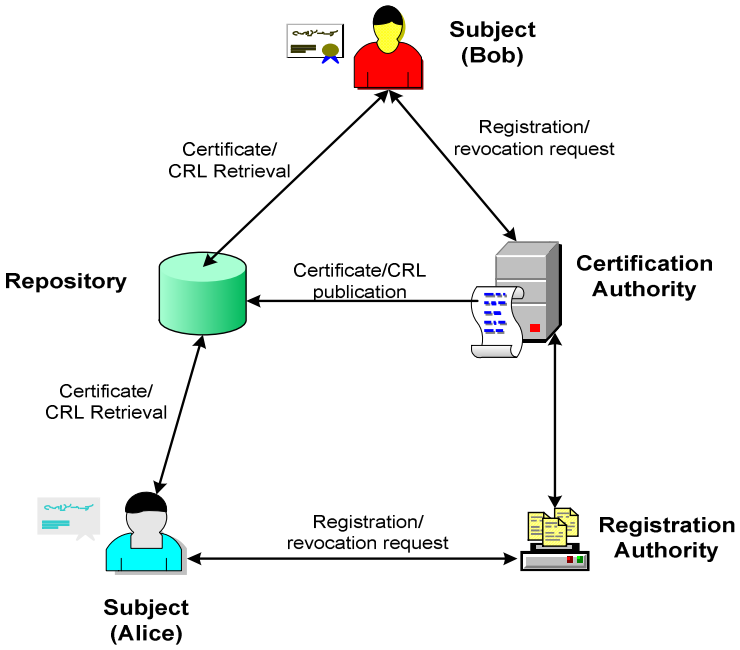


Figure 3.7: Components of a PKI

Registration is the first step for a subject to apply for a certificate. This process could be directly accomplished by the certificate authority or through the delegated registration authority (for example, as shown in Figure 3.7, Bob's registration is directly handled by the certificate authority, while Alice registers herself at the registration authority). Once the identity of the subject is validated and the possession of the private key corresponding to the public key is verified, the CA and only CA in a PKI will issue the certificate for that subject. The issued certificates are stored and published on an X.500-based directory. A subject can retrieve its own certificate or other subject's certificates from the directory when needed, using the Lightweight Directory Access Protocol (LDAP), or the File Transfer Protocol (FTP), or the Hyper Text Transfer Protocol (HTTP). The certificate revocation is an important function in the PKI. There are a lot of good reasons for a subject to cancel

its previously issued certificate such as the expiration of the certificate, compromise of the private key, change of affiliation etc. After receiving the revocation request from a subject the certificate authority publishes the related certificate revocation information about that subject via CRL, which is also deposited at the X.500-based directory, so that any user can check the revocation status of a certificate. As shown above, the certificate authority plays the crucial role in a PKI, since no components except the certificate authority can issue the certificates and CRLs.

PKIs possess diverse architectures due to the different requirements of practical applications. A PKI might contain only one certificate authority. In some cases a PKI might be made up of many certificate authorities that are arranged either in a hierarchical or in a peer-to-peer architecture.

Single CA

The basic PKI architecture is one in which a single CA is responsible for the management of all certificates. Figure 3.8 depicts a PKI with single CA.

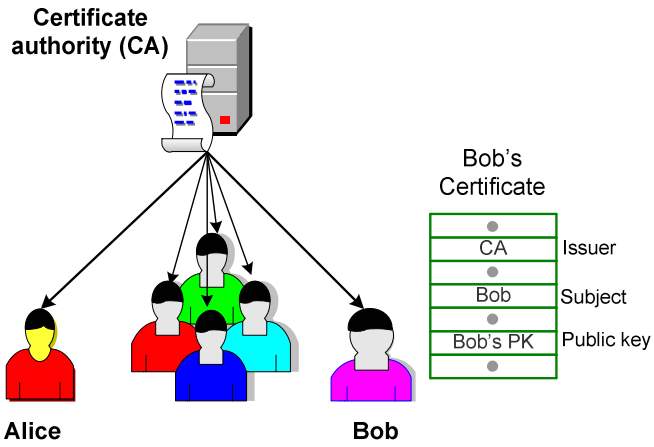


Figure 3.8: A PKI with single CA

As shown in Figure 3.8, all the users commonly place their trust on that single CA. They assure that the CA issues each certificate only after successfully verifying the identity and the associated public key of an applicant using the standard registration procedure. In this simple architecture it is assumed that the public key of the CA is securely distributed to everyone in an out-of-band manner. As shown in Figure 3.8, Alice can acquire Bob's right public key by validating the signature in Bob's certificate using the public key of the CA. There are two means for Alice to obtain Bob's certificate: retrieving from the publicly accessible repository, or directly receiving from Bob. However, this simple scheme does not scale up well, because it is difficult to support the certificate management in a large organization such as an international enterprise whose branches are spread across the globe. It is

impractical that the certificate management is concentrated in a single authority for a big international organization. The natural solution to this problem is that each branch could locally set up its own CA responsible for the certificate management for users in that branch. These CAs further are organized in a hierarchical fashion.

Hierarchical architecture

The hierarchical tree PKI is used to address the scalable issue of the certificate management in a large organization. It usually contains a number of CAs. To arrange them in a hierarchy, the trust relationship between two CAs has to be established. This is realized in the way that a CA issues a certificate to another CA. Such certificate is referred to as a *cross-certificate*. At the top of the hierarchy is the root CA called the *trust anchor* which is the single point that all entities (i.e. users, subordinate CAs) within the PKI commonly trust. Its public key is well-known to every entity. The root CA issues cross-certificates to its subordinate CAs. These CAs in turn issue cross-certificates to their subordinates CAs, or certificates to users (see Figure 3.9).

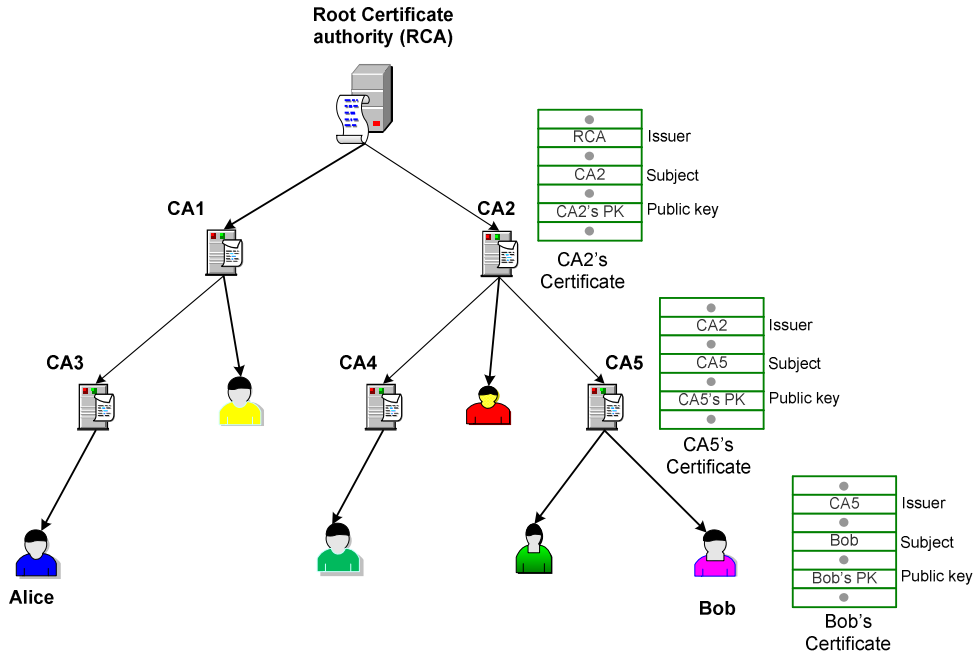


Figure 3.9: Hierarchical PKI

The certificates including cross-certificates are chained to form a *certification path*. For example, in Figure 3.9, the certification path from the root CA to user Bob is: root CA→CA2→CA5→Bob, where X→Y means that X issues a certificate to Y. To validate another user's public key, the public key of the root CA is first applied to verifying the authenticity of its subordinate CA's certificate in order to extract the certified public key of that subordinate CA. Then this extracted public key is utilized to verify the certificate of

next subordinate CA. This certificate validation is recursively processed along the certificate path until the target partner is reached. As a result of this process, the validity of the desired partner's public key contained in its certificate is confirmed. Figure 3.9 illustrates a scenario how Alice checks the validation of Bob's public key.

The original design of X.509 is intended to use a single hierarchical PKI to support the certificate services for the whole world. However, such assumption has not come to reality yet and maybe never realizes in future. It is impossible that all organizations in the world place their trust on a single root CA, because each organization has its own interests and wants to control over its own organization as maximum as possible. Thus each organization will establish its own PKI. To make a secure communication between two organizations, a trust relationship between two PKIs has to be established. This introduces the third kind of PKI architecture, i.e. the peer-to-peer architecture.

Peer-to-peer architecture

The peer-to-peer PKI architecture bridges the communication between organizations (see Figure 3.10).

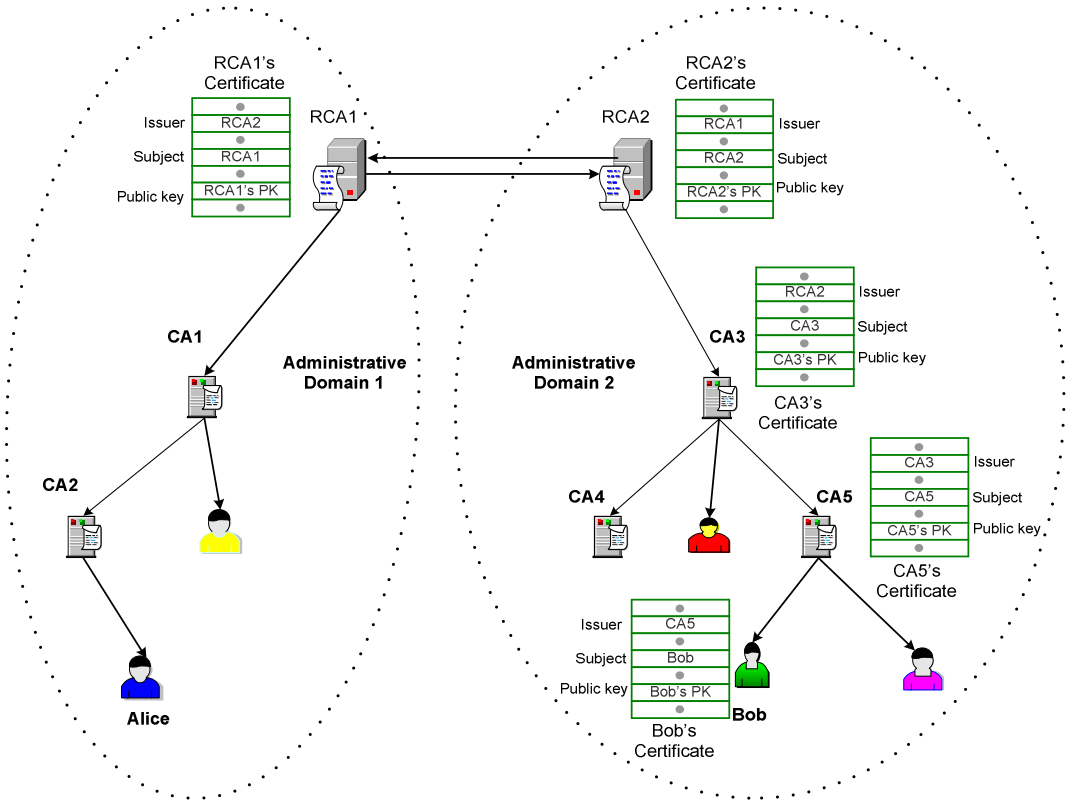


Figure 3.10: Peer-to-peer certification architecture

Each organization has its own administrative domain for the certificate management, where all users still put their trust only on their local root CA like in a hierarchical PKI. To make the secure communication between two organizations possible, the trust relationship between them has to be established. For this, two root CAs issue certificates to each other, i.e. *mutual cross-certification*. It is worth noting that this is different from the *unidirectional cross-certification* in a hierarchical PKI, where only a superior CA can issue a cross-certificate to a subordinate CA and the reverse direction for the certificate issue is prohibited. To verify the public key from another administrative domain, the first step is to validate the cross-certificate of remote root CA using the public key of the local root CA in order to extract the correct public key of remote root CA. Once this public key is obtained, the same procedure used for certificate validation as described in the hierarchical PKI is executed to verify the public key of the target partner within the remote administrative domain. As shown in Figure 3.10, Alice residing in administrative domain 1 takes the following steps to get the correct public key of Bob in administrative domain 2: public key of RCA1→RCA2'certificate→CA3'certificate→CA5'certificate→Bob's certificate, where → denotes a certificate validation operation. Similarly, Bob can acquire the correct public key of Alice since his root CA (RCA2) has already issued a cross-certificate to the root CA of Alice (RCA1).

3.4.2 PGP certificate management

The PGP certificate significantly differs from the X.509 certificate in many aspects regarding the management such as certificate issue, trust management, certificate revocation. PGP certificates are issued by users themselves rather than by a commonly trusted certificate authority (CA) like X.509 certificates. Each user signs the public keys for the people he/she acquaints. PGP employs introducer mechanisms to establish a secure communication between a user and the people he/she never met before. As shown in Figure 3.11, Bob knows Carol well and ensures the authenticity of Carol's public key. Bob now wants to communicate with Anan with whom he previously has not had any interactions. While Carol is acquainted with Anan, and Carol has signed Anan's public key in a certificate. Carol as an introducer forwards this certificate to Bob. Bob can verify it using Carol's public key. The successful verification makes Bob confident that Anan's public key is authentic. In this way each user can gradually form a web of public keys linked by the certificates.

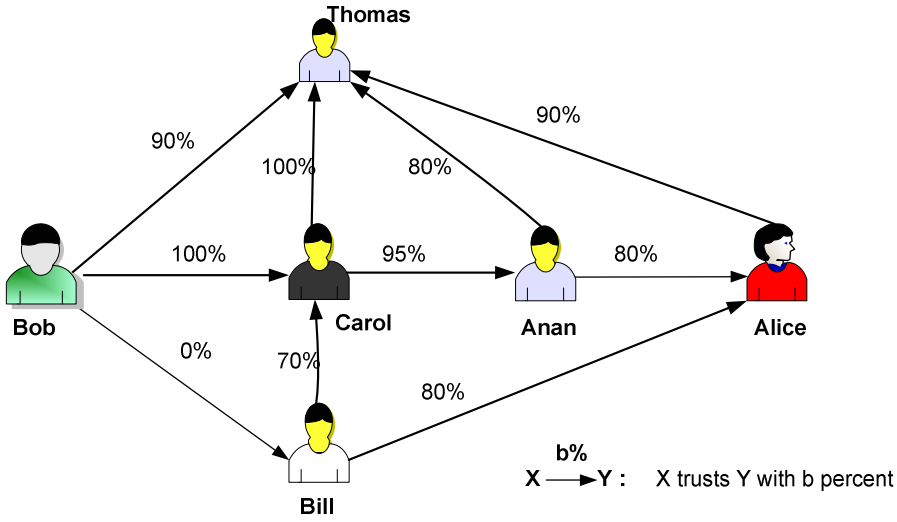


Figure 3.11: Example of web of trust

X.509 certificates are managed in a centralized public key infrastructure (PKI). Each user completely trusts the root certificate authority. This single point of trust model simplifies the trust management of public keys. As long as the validation of a certificate is successful, a user can fully trust the authenticity of the public key contained in that certificate. PGP adopts web of trust approach to establish the different level of trust to a user's public key which is a decentralized certificate management approach [75]. Corresponding to the relationship between people in the real world, a user in PGP assigns unequal trust levels to the different introducers. He/she may fully, or marginally, untrustworthily trust an introducer to issue certificates to other users. As shown in Figure 3.11 Bob assigns Carol, Thomas, and Bill with trust values of 90%, 100%, and 0% respectively. As a result, besides the verification of the signatures, a user has to calculate the trust value of the target user's public key along the certificate paths, in order to determine to what extent he trusts that public key. From each user's perspective, the web of trust linked by certificates forms a directed graph, where a node represents a user, a directed edge between two nodes represents direct trust relation between two users, a weight on a directed edge indicates to what degree the starting node of this edge believe the end node to sign other's public keys (see Figure 3.11). It is not a trivial thing to calculate the trust value between any two nodes in a directed graph since there maybe exists multiple paths between them. Several schemes have been proposed to address this problem [76], [77]. Taking a simple example shown in Figure 3.11, two paths exist from Bob to Alice when computing the trust value of Alice. Fortunately, one path bypassing Bill is invalid in this sample since Bob does not trust Bill at all. So Bob can simply get the trust value of Alice by multiplying the separate trust value along the trust path (i.e., Bob→Carol→Anan→Alice). The resulting value is 76%, which means that Bob

trusts Alice's public key in its certificate with 76% trustworthy, even if all certificates along the trust path is validated.

In PKI, the certificate revocation is centrally handled by the certificate authority (CA) so that all users can quickly know the status of other certificates by querying the repository. Whereas in PGP it is accomplished by that the owner issues a key revocation certificate signed by him. After that, the owner does his best to disseminate this certificate as widely as possible, to notify the other users about the invalidity of his public key.

3.4.3 Comparisons between two approaches

As introduced above X.509 certificates and PGP certificates both can be applied to the secure dissemination of a public key. In the following, the comparisons between them in aspect of their relation to the P2P model and in relation to security are given when they are deployed in a P2P application.

Relation to P2P model

PGP certificates are solely managed by users themselves rather than relying on a central CA. This distributed nature makes them pretty suitable for these applications set up by using pure P2P model, where no infrastructure is available for the service support at all time. Some PGP-like certificate management schemes have been proposed for the use in pure P2P applications [78], [79]. In contrast, X.509 certificates completely rely on a public key infrastructure (PKI) responsible for their management. To establish a PKI, the central certificate authority (CA) has to be introduced, since most of the certificate managements are basically governed by it. It provides off-line certificate services for its users, e.g. certificate issue, certificate revocation. Users do not need the CA to issue a certificate for each communication. Basically, once a certificate is issued it remains valid until its expiry. As a result, users can communicate to each other directly without the help of the CA except the bootstrapping phase of the system. Therefore the PKI quite well matches the hybrid P2P model, where some dedicated servers are used to assist users to set up the communication, but the communication runs directly between users without passing these servers. Several hybrid P2P applications have deployed X.509 certificates for authentication and confidential services [80], [81].

Security

In general, without trusted centralized administration it is very difficult to achieve a high level security for the certificate management. Decentralized managed PGP certificates are therefore susceptible to the following security weaknesses compared to centralized managed X.509 certificates.

- **Sybil attacks** [44]: A user of PGP certificates may introduce him/her to different partners with different names, since there is no rigid registration procedure like in X.509 certificates. An example of such attacks in a video conference is that an attacker can use different names to attend a conference so that he/she may always be present in that conference although he/she has never been invited.
- **Authenticity of a public key**: PGP exploits the web-of-trust approach to evaluate the trust level of a public key. The resulting trust value can only provide the probabilistic guarantee of the authenticity of a public key. This is significantly different from the centralized managed X.509 certificates, where a user can completely assure that the public key is really bound to the subject name in a certificate once the signature is verified.
- **Malicious introducers**: A malicious introducer can compromise the security of a PGP user community either by introducing bad guys or by a false binding of a public key to the name deliberately.
- **Revocation**: The consistent status of a certificate is difficult to achieve for a user community, because there is no central place to store the certificate revocation lists like in X.509 certificate.

To sum up, analogous to paper certificates in the real world, X.509 certificates look like official paper certificates (e.g. passport), while a PGP certificate is at most equivalent to a recommendation letter in terms of functionality. X.509 certificates are best suitable for the business communication, where there are strict security requirements. PGP certificates are considered unreliable for such kind of communication, because PGP has no official mechanisms for the creation, acquisition, and distribution of certificates. PGP is merely appropriate for a private communication which usually has relatively loose security requirements [82]. The object of this thesis is to design a secure P2P video conference system used in enterprise environments to provide secure video conference services among the people who might never met before. Thus, X.509 certificates should be deployed in the system to meet these rigid security requirements required by an enterprise. The X.509 certificates deployment implicates that we have to apply the hybrid P2P model to our system, because a central CA is involved. This is another important reason that we build our system using the hybrid P2P model.

Chapter 4

Use of Secure VPNs for P2P Conferences

Nowadays secure virtual private networks (VPNs) have been massively deployed in enterprise environments to protect various applications, in particular client/server applications across public networks. Virtual private networks are considered as one of the most matured security architectures in use. In this chapter we discuss their use to support P2P conferences in terms of security, flexibility, and efficiency.

4.1 Requirements to security architectures of P2P conferences

Security architectures are frameworks which specify how to incorporate the necessary cryptographic methodologies and security functions (e.g. key management) into the system to meet the defined security. To secure P2P conferences security architectures should at least support the known basic security demands: *confidentiality*, *integrity*, *user authentication*, and *authorization*. Due to their decentralized structure and their real-time requirements P2P conferences should further meet the following requirements:

End-to-end security

Usually two kinds of security services can be offered in an enterprise network: end-to-end security, or site-to-site and site-to-end security, respectively. The so-called end-to-end security means that messages are securely delivered from the sender' host to the receiver's host and that they are not accessible to any intermediate node or server along the transmission path. Site-to-site and site-to-end security mean that messages are merely protected during wide area network (WAN) transmission, while they are transmitted in the plaintext form within the site scope.

It is obvious that P2P conferences have to apply end-to-end security for several reasons. (1) Security threats occur not only during WAN transmission but also at local site as indicated in [83]. A significant number of threats originate from insiders. (2) In order to protect en-

terprise business secrets, enterprises demand that business information should be only accessible to authorized group members but not to people outside the group, even if they belong to the same enterprise.

Group key management

In a secure P2P conference more than two participants are usually involved. A group key management protocol rather than a two-party key exchange protocol has to be applied to securing the group communication. Two-party key exchange protocols are inefficient for group communication, because each member has to negotiate an individual key with the other group members. Each message sent to the group has to be separately encrypted with the respective keys of the group members, i.e. *n-1* encryptions are required. On the contrary, only one encryption is needed for forwarding a message to the group when a group key management protocol is employed (see Figure 4.1).

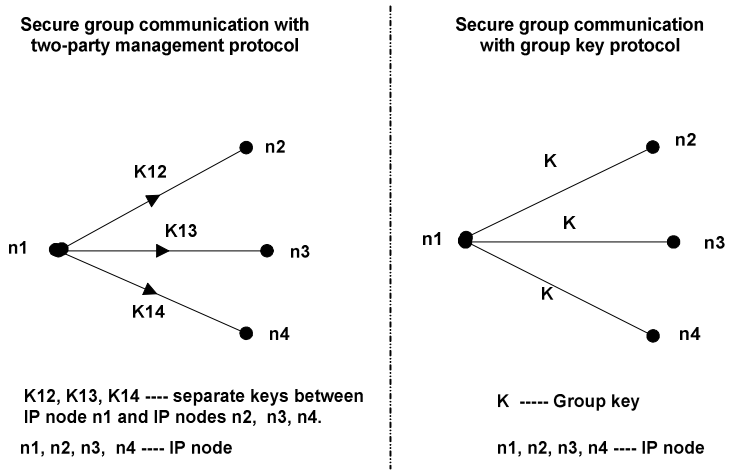


Figure 4.1: Group key versus two-party key exchange

Flexible security policy enforcement

The security policy determines the desired protection level of a conference and specifies the security algorithms to be applied. The security policy of a P2P conference should be determined by the participants themselves rather than by a dedicated network administrator when running the conference, since a P2P conference is autonomous and consists of a transient group. The applied policy should be allowed to be attuned in the course of the conference to provide more flexibility for users.

Efficiency

Security always imposes additional processing burdens on the system. These burdens may pose a negative impact on the quality of service (QoS). For example, a secure conference

incurs longer end-to-end communication delays due to message encryption/decryption. Therefore, the deployed algorithms and protocols should be efficient enough to meet the strict QoS requirements of real-time communication.

4.2 VPNs

A virtual private network (VPN) is a private data network that makes use of the public telecommunication infrastructure (e.g. the Internet) to establishing private connections between group members either by individually applying tunneling technologies or security procedures (such as encryption, authentication), or by using both technologies at the same time [84], [85].

Network tunneling is intended to establish a logically private connection over a public network. A tunnel is constructed by imposing an extra header to the original packet which identifies it as VPN traffic. This newly constructed packet is forwarded by intermediate nodes based on this outer extra header without looking up the header of the original packet in the public network. At the boundary of the VPN the extra header is stripped off and the packet is forwarded to the intended destination according to the original header. Tunneling technologies only encapsulate a packet by means of an extra header. They do not touch the payload of the packet. This means that the message in a packet is transmitted in clear form so that the term *private* in VPN technologies cannot be equally viewed as the term *secure.* VPNs which are set up by merely using tunneling technologies, such as ATM/FR VPN [86], layer 2 MPLS VPN [87], and layer 3 MPLS VPN [88], usually do not give any security guarantees due to the absence of built-in security functions. They only provide the private data transmission by correctly delivering data to the destination. This is achieved by separating traffic within a VPN through examining the extra header of a packet.

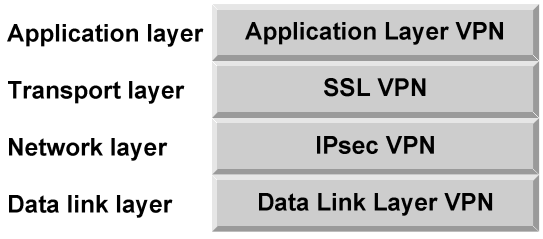


Figure 4.2: Secure VPNs

In contrast to that, VPNs which are set up by using security procedures can secure the data transmission. This kind of VPNs is therefore regarded as *secure VPNs*. They are our main concern here. In secure VPNs security functions can be introduced at different levels. Related to the TCP/IP protocol stack there exist four kinds of secure VPNs: data link layer

VPNs, IPsec VPNs, SSL VPNs, and application layer VPNs (see Figure 4.2). In the sequel we assess the strengths and weaknesses of the different VPNs for secure P2P conferences according to the security requirements introduced above.

4.2.1 Data link layer VPN

Data link layer VPNs can be established using one of three protocols released by IETF: *Point-to-Point Tunneling Protocol* (PPTP) [89], *Layer 2 Forwarding* (L2F) [90], and *Layer 2 Tunneling Protocol* (L2TP) [91]. L2TP was intended to replace PPTP and L2F, because it combines the best features of PPTP and L2F. In the traditional corporate dial-in solution, remote employees need to place long-distance calls to the Network Access Sever (NAS) for their access to the corporate network. L2TP splits the NAS into two physically independent devices: the *L2TP Access Concentrator* (LAC), which is located at Internet service provider (ISP) to provide the physical connection to the dial-in user or directly embedded in remote clients, and the *L2TP Network Server* (LNS), which acts as a gateway to the corporate network. The LAC and LNS communicate via an L2TP tunnel. In the L2TP compulsory tunnel mode an L2TP tunnel is created in two steps (see Figure 4.3). First a PPP frame from the remote client is added an L2TP header to form an L2TP frame. Then this frame is encapsulated inside a UDP packet, which in turn is encapsulated inside an IP packet whose source and destination addresses define the L2TP tunnel's endpoint. The voluntary tunnel mode is another operation mode of the L2TP protocol, where the L2TP LAC functionality is built in the remote client so that ISPs are not involved in setting up L2TP tunnels. This mode applies the same principle as the compulsory tunnel mode. The only distinction between two modes is whether the client is able to establish an L2TP tunnel bypassing the internet service providers or not.

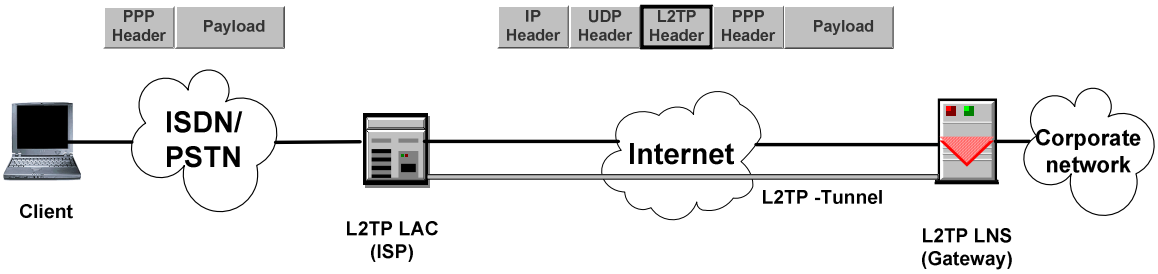


Figure 4.3: L2TP compulsory tunnel model

Strength and weaknesses in a P2P conference

Due to the absence of built-in security functions in L2TP data payloads are transmitted as clear text via the L2TP tunnel. To address this problem IETF has ratified RFC 3193 [92], which combines L2TP with IPsec, to assure the basic security demands mentioned in Sec-

tion 4.1. L2TP is commonly applied to dial-up communications between a mobile user and the gateway of its enterprise network. So it provides site-to-end security rather than end-to-end security when a P2P conference running on top of it.

4.2.2 IPsec VPNs

IPsec VPNs are enterprise networks which are deployed on a shared infrastructure using the IPsec technology. IPsec is an open, standard-based security architecture defined by a series of standards (RFC 2401-2412, 2451). It was designed to protect traffic between two IP nodes at the network level assuring data integrity, authenticity, confidentiality, and anti-replay. IPsec supports two security protocols: *IP Authentication Header* (AH) [93] and *IP Encapsulating Security Payload* (ESP) [94]. The difference between the two protocols is that the latter supports all security services mentioned above, while the former does not assure confidentiality that most applications need. In [95] therefore it was suggested that AH protocol ought to be abolished due to its overlapping features with ESP. Both protocols can operate either in transport or tunnel mode. In transport mode the security protocol header is inserted between the IP header and the upper layer protocol header. Thus this mode offers security services only to the IP payload not to the whole IP packets. The transport mode is intended for end-to-end protection between two hosts. In tunnel mode the original packet is wrapped in a new IP packet by adding a new IP header. It provides protection to the entire IP packet because both the header and the payload of the original packet are viewed as the payload of that new IP packet. The secure path protected by this mode may be only a fraction of the end-to-end path between the source and destination hosts of the original header, since the new IP header is different from the original IP header. For this reason, the tunnel mode is typically used between security gateways or between a host and a security gateway.

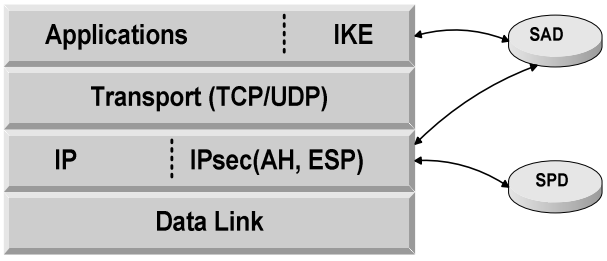


Figure 4.4: IPsec Architecture

The IPsec architecture makes use of the following three auxiliary components to support the AH and ESP protocol (see Figure 4.4): Security Policy Database (SPD), Security Associations Database (SAD) and Internet Key Exchange protocol (IKE).

SPD

The *Security Policy Database* (SPD) contains security policies manually configured by the system manager. These security policies specify what security services are provided to a packet and in what fashion they are applied. When a packet arrives the SPD is first used by the IP node to determine what kind of actions should be applied. There are three possible actions:

- *Discarding:* The packet is dropped.
- *Bypassing*: The packet is relayed without applying IPsec.
- *Protecting*: The packet is processed by IPsec.

Apart from specifying which of the above mentioned actions is taken, each policy also describes the mode, the algorithms and the protocols to be applied if IPsec processing is required. The policies in the SPD are sorted. Each policy is indexed by a selector. The selectors are extracted from the network and transport layer headers. The following fields can be used as a selector: source address, destination address, name, transport layer protocol, and upper layer ports.

SAD

A *security association* (SA) is an agreement on a set of parameters needed for secure communication between two IP nodes, which includes the security protocols (AH, ESP, or the combination of both), the cryptographic algorithms, the keys to be used, and the lifetime of the security association. The SAs are stored in the *Security Associations Database* (SAD). An entry in the SAD is uniquely identified by a triple consisting of a Security Parameter Index (SPI), an IP Destination, and a security protocol (AH or ESP) identifier. The SPI is transmitted inside the AH or the ESP header. When the packet arrives at the destination the IP node can choose the right SA to be applied for decrypting and/or authenticating the packet using the SPI value. An IP node may have stored several security associations in its database to securely communicate with many other IP nodes or to protect different kinds of traffic between it and another IP node. The security associations can be manually configured by the system manager before the deployment or automatically established by using the internet key exchange protocol (IKE) [96] at runtime.

IKE

The IKE protocol, which runs at the application layer, is the foundation of the IPsec architecture. It is used to authenticate the communication partners, to establish a security association, and to provide a key management service (generating and refreshing session keys). IKE consists of two phases. Phase (1) comprises the mutual authentication of the two IP nodes and the establishment of a secure channel between them. The mutual authentication

can be performed using digital signatures or pre-shared secrets. In phase (2) the two IP nodes agree upon a shared session key and the common security associations using the secure channel established in phase (1). The two nodes can restart phase (2) as long as the renewal of the session key is required.

Strength and weaknesses in a P2P conference

The most important advantage of IPsec is that it is transparent to applications. Any IP based application can get total protection when it is deployed. Moreover, the upper layer applications do not need any modifications for their security requirements. However, several disadvantages inherently exist when IPsec is used for a P2P conference.

- *Inflexible security policy enforcement*
  Prior to the deployment of an IPsec VPN, the associated security policies (i.e. SPD) must be manually configured in the related IP nodes. This specific task is usually only allowed for the network administrator but not for the general user, because IPsec is implemented in the kernel [97]. This implies that IPsec VPNs can only enforce a static security policy, which remains unchanged for a quite long time, rather than dynamic security policies, which could be flexibly set by the users themselves during a P2P conference.

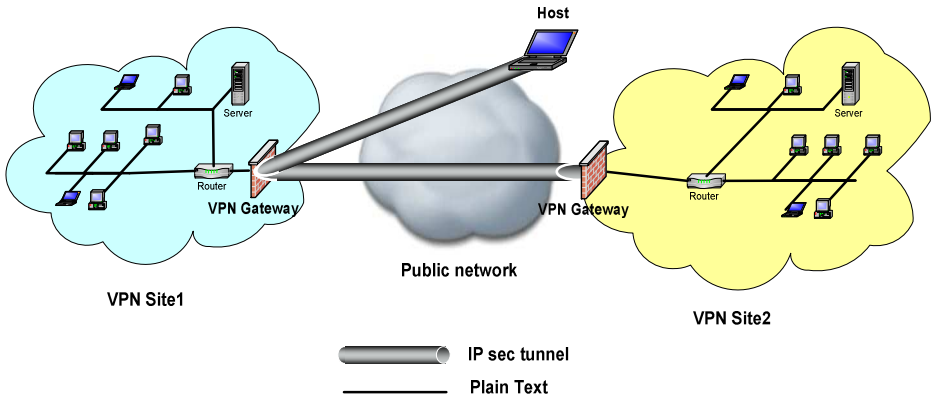


Figure 4.5: Gateway-to-gateway and host-to-gateway architecture

- *Difficult to offer end-to-end security*
  IPsec VPNs operate in the network layer which is the lowest layer to provide end-to-end security in theory, but in practice IPsec VPN rarely adopts host-to-host architecture to provide end-to-end security for data transmission. This is because the configurations (e.g. security policy enforcement) on each host involved in the VPN have to be manually carried out by the network administrator. This is an unbearable burden for the system administrator, especially when the number of users is large [98]. Therefore IPsec VPNs tend to prefer host-to-gateway or gateway-to-gateway architectures to provide

end-to-site and site-to-site security, respectively. Figure 4.5 illustrates a scenario of the gateway-to-gateway and host-to-gateway architecture of IPsec VPNs.

- *Inefficient group communication*
  Currently IPsec does not support a group key management but only a two-party key management.

4.2.3 SSL VPNs

SSL VPNs are based on the commonly used protocol SSL (*Secure Socket Layer*) for secure data transmissions over the Internet. SSL operates on top of TCP to provide reliable end-to-end security services for applications. It was standardized by IETF where it is called TLS (*Transport Layer Security*) [20]. It consists of four sub-protocols as illustrated in Figure 4.6.

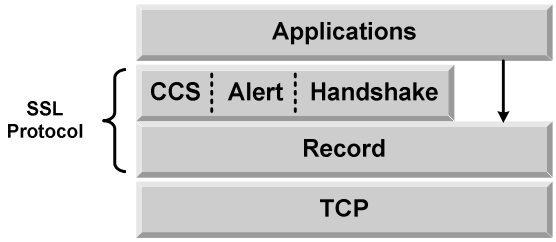


Figure 4.6: SSL protocol stack

The four sub-protocols are separately introduced in the following:

- *Change Cipher Spec Protocol* (CCS) is used to notify the *Record protocol* about the change of security parameters, so that both end points can update the cipher suite used on the established connection.

- *Alert protocol* is triggered when errors occur or to tear down sessions when they are completed.

- *Record protocol* applies all security parameters (including the session key) agreed in the handshake protocol to provide data confidentiality and integrity services for upper layer applications.

- *Handshake protocol* is used to agree upon a common cipher suite used during data transfer, establish a shared session key between the client (i.e. browser) and server, compulsorily authenticate the server to the client, and optionally the client to the server:

- *Server authentication*: Upon receipt of the connection request from the client, the server sends its X.509 certificate to the client, which has been issued by a certificate authority (CA) listed in the client's list of trusted CAs (usually preinstalled in the browser). After checking the validity of the server's certificate, the client creates a *pre_master_key* and transmits it to the server by encrypting with the server's public key extracted from the validated server certificate. The server and client meanwhile generate the *master_secret* with the *pre_master_key*. Finally, the server authenticates itself to the client by sending the *Finished* message which is a hash of all the messages that the server sent and the shared *master_secret.*

- *Client authentication*: When the client receives the authentication request from the server, it first responds with its X.509 certificate. Next it creates a signature using the *master_secret* and all exchanged messages and delivers it to the server. The server authenticates the client by correctly verifying the signature using the public key extracted from the client's certificate.

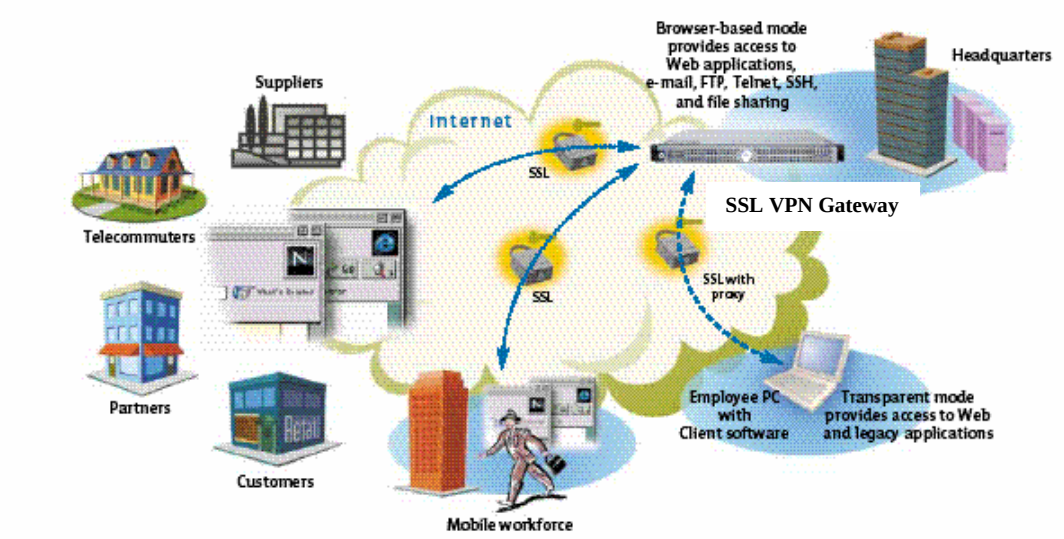


Figure 4.7: SSL VPN provides secure remote access to corporate networks [99]

SSL is extensively used in web applications to provide end-to-end protection between the browser and web server. However, VPNs based on SSL technology serve as remote access VPNs to provide the end-to-site security between remote users and the SSL VPN gateway residing in corporate networks as shown in Figure 4.7. SSL VPNs at least have the following two advantages over IPsec VPNs when they are used for the remote access to corporate networks:

- *Granular access control*: SSL VPNs can support user-level authentication ensuring that only authorized users have access to the specific resources as allowed by the company's security policy.

- *Lower cost of ownership*: SSL VPNs do not require special client side software since broadly used web browsers, such as Internet Explorer and Netscape Navigator, contain the SSL function. On the contrary, IPsec VPN requires special client software operating at the client side.

Strength and weakness in a P2P conference

Compared to IPsec VPNs the advantage of SSL VPNs is that they can be deployed without updating the operating system, because they are implemented outside the kernel. Like IPsec VPNs there are problems for SSL VPNs to be used in P2P conferences:

- *Inefficient group communication*
  This is simply because the handshake protocol deals with the key management only for two parties rather than for the whole group.

- *Rarely used for the end-to-end protection*
  SSL VPNs can inherently be used in a host-to-host fashion to provide end-to-end security, but in practice they are mostly deployed for the remote access to support end-to-site security.

- *Exclusive support of TCP-based applications*
  SSL merely supports TCP based applications because its design assumes that the underlying layer offers a reliable transport service. UDP is an efficient but unreliable transport protocol which does not handle packet losses problem. If SSL would be used in UDP based applications, then packet losses are viewed as security breaks that force to disconnect the communication [20]. Most real-time applications, however, including P2P conferences run over UDP for the efficiency reasons. This limitation precludes the use of SSL VPNs in a P2P conference.

4.2.4 Application layer VPNs

Application layer VPNs utilizes the security functions embedded in the related applications. PGP [64] for secure E-mail and Groove [34] for secure collaborative workspace are typical examples of application layer VPNs. Actually, these VPNs act as instant VPNs, i.e. when the VPN service (e.g. secure e-mail) is needed then the related VPN will be created. It disappears when the service terminates. No special administration support for configuring the VPN is required. All operations and configurations are defined by the participants them-

selves when invoking the service. Due to its embedded implementation it can provide a more tailored protection compared to underlying layer VPN technologies. Different security measures can be taken corresponding to the type of the application. Moreover, appropriate security algorithms and protocols such as a group key management protocol can be readily integrated to meet the security demands mentioned in Section 4.1. The major drawback of application layer VPNs is that some modifications have to be made in the applications to add these security functions. The designed security architecture is solely available for the targeted application. There is no general scheme at this layer suitable for the diverse applications. For example, the security architecture of PGP is impossible to be used for Groove.

4.2.5 Summary

The comparison of the four kinds of VPN technologies is summarized in Table 4.1.

Table 4.1: Comparison of VPN technologies

Security requirements	Date link layer VPN	IPsec VPN	SSL VPN	Application layer VPN
Basic security services	Yes (With IPsec)	Yes	Yes	Yes
End-to-end security	No	Difficult	Yes (Rarely used)	Yes
Group key management	No	No	No	Yes
Flexible security policy enforcement	No	No	Difficult	Yes
Supporting TCP and UDP-based applications simultaneously	Yes	Yes	TCP only	Yes
Transparent to applications	Yes	Yes	Yes	No

It shows that data link layer VPNs and SSL VPNs are inappropriate for P2P conferences, because the first one does not provide end-to-end security while the latter does not support UDP based applications. A straightforward solution would be the direct use of existing IPsec VPN infrastructures to support a P2P conference. Unfortunately, IPsec VPN infrastructures are mostly established using gateway-to-host or gateway-to-gateway architectures rather than host-to-host architectures. Thus there is a gap between security capabilities of the VPN and the end-to-end security requirements of the P2P conference. Furthermore, a missing group key management and inflexible security policy enforcement make it difficult for IPsec VPNs to supporting a dynamic and efficient P2P group communication. To fully meet the security requirements of a P2P conference the design of dedicated security architectures seems the most appropriate way, even it is more costly.

The security architecture specially designed for P2P conference systems at the application level brings not only the security advantages but also the usability advantages. This archi-

tecture is independent of the underlying network security infrastructures. So users can spontaneously hold a secure P2P conference without caring about whether networks are secure or not. In addition, the management cost for setting up secure P2P conferences can be drastically reduced or fully eliminated since the network managers do not need to configure security parameters for these conferences. However, designing a security architecture is not a trivial thing. The next chapter is devoted to addressing this issue.

Chapter 5

A Security Architecture for the BRAVIS System

As shown in the previous chapter, a specific security architecture designed at the application level is strongly demanded to protect P2P video conferences. This chapter is targeted towards the development of such security architecture and illustrates how it works in a P2P video conference system using the BRAVIS system as an example.

5.1 General design considerations

Security architecture for client/server conference systems

Maybe for similar reasons as discussed in the previous chapter, most client/server based video conference systems have positioned their security architecture at the application level as well. The H.323 systems, one of widely used client/server based video conference systems, have a matured security architecture which has been specified in the H.235 standard [23]. The design of the H.235 architecture follows the client/server model. The gatekeeper and the MCU are used not only to support the conference control but also to provide some necessary security services for running secure video conferences. The gatekeeper is responsible for user authentication and access control when a user wants to join the conference. The generation, distribution, and update of the group key are accomplished by the MCU during the conference. Like any other client/server architecture this security architecture is vulnerable to single point of failure or performance bottleneck. Moreover, the gatekeeper and the MCU present attractive points for attacks. The security of H.323 video conferences mainly relies on the trustworthiness of the gatekeeper and the MCU. Once they are compromised, all video conferences which are running on top of them are completely exposed. The attractive feature of the H.235 architecture is that it is relatively easy to implement, since most important security functions are centrally executed in the servers.

Security architecture for P2P conference systems

P2P video conferencing is a newly emerging application in the Internet. No standard has devoted to addressing the security issues of a P2P video conference as yet. Designing the security architecture for a P2P videoconference is not a trivial thing because the setting in a P2P system is significantly different from that in a client/server system. Like any other P2P system designs, the security architecture may be devised in a pure or hybrid fashion. However, we cannot design a pure P2P security architecture for P2P conference systems considering the possible attacks such as Sybil attacks as discussed in chapter 3. There is no central control in a pure P2P system. Security related management functions such as identity management and public key management are decentralized. No central authority is responsible for these tasks. This allows an attacker to use different identities to attend conferences (i.e. Sybil attacks). To address this problem, we have to design a hybrid P2P security architecture for a P2P conference system in which a certificate authority (CA) is introduced to centrally control the identities and the public keys of the participants, whereas the other security functions necessary for a secure conference such as user authentication, authorization, and group key management are moved to users. The CA issues a certificate signed with its private key for every possible participant which binds the identity and the public key of the user. The identity is distinguished information of each user like its E-mail address. Certainly, participants do not need the CA to issue a certificate for each communication. Basically, once a certificate is issued it remains valid until its expiry.

Despite the use of the central server CA the hybrid P2P security architecture does not suffer from the known weaknesses in client/server systems. This is because the CA only provides off-line security services (i.e. issuing certificates to users), and is not involved in each secure group communication at all. Moreover, the hybrid P2P security architecture is more robust to attacks than the H.235 architecture. There are no central servers responsible for security services management for running conferences. Thus when using a hybrid P2P architecture, attackers have to separately intrude each conference if they want to access the contents of all conferences, whereas this requires attackers to only compromise two servers in the H.235 architecture.

In the following section we take the BRAVIS system as an example to show how to build a security architecture for a P2P conference system. The operation of a CA will not be discussed below, and we simply assume that it is already available.

5.2 A security architecture for the BRAVIS system

The security architecture specifically designed for the BRAVIS system [182] is a hybrid P2P security architecture in which peers themselves manage security services and each peer

possesses the identical structure as shown in Figure 5.1. This architecture protects meetings on an end-to-end basis by providing security services including user authentication, authorization, group key management, security policy management, and data confidentiality/integrity. Several modules are designed to offer these security services. Each module merely implements one dedicated security function for easing the system maintenance and expansion. The modules are grouped together and form a security layer. It is embedded into the BRAVIS system, and placed between the application and communication layer. The security layer should not be placed beneath the communication layer. There are at least two reasons for this. First messages originating from the security layer have to rely on the underlying group communication protocol for an ordered and reliable message delivery. Second the RTP (Real-time transport protocol) header should not be encrypted so that RTP header can be compressed for the bandwidth reduction [23]. The security modules of the security layer are introduced in the sequel.

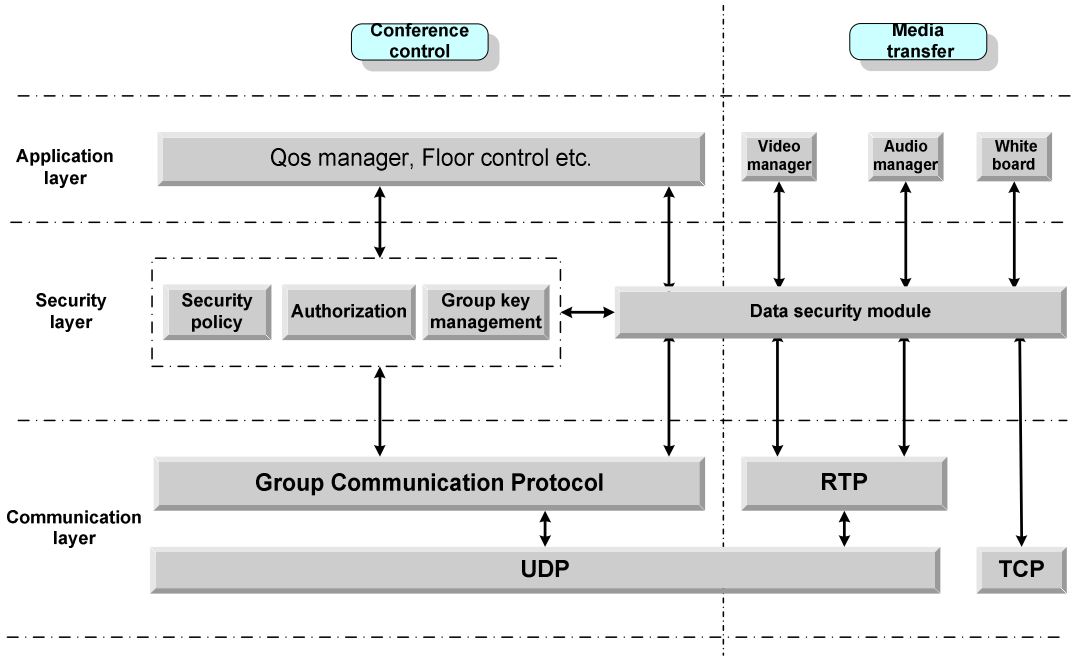


Figure 5.1: Secure BRAVIS system

5.2.1 Security policy module

The policy module decides which security level and what kind of security algorithm are enforced in the conference. The main reason to designing a conference system with diverse security levels is that different conferences may place unequal security requirements on the system. For example, a business meeting needs high level security, whilst a teleseminar

may not require any protection at all. Moreover, devices used by participants may have different processing capabilities so that the participants have to negotiate an appropriate security policy to enroll the participant with a lower power device into the meeting.

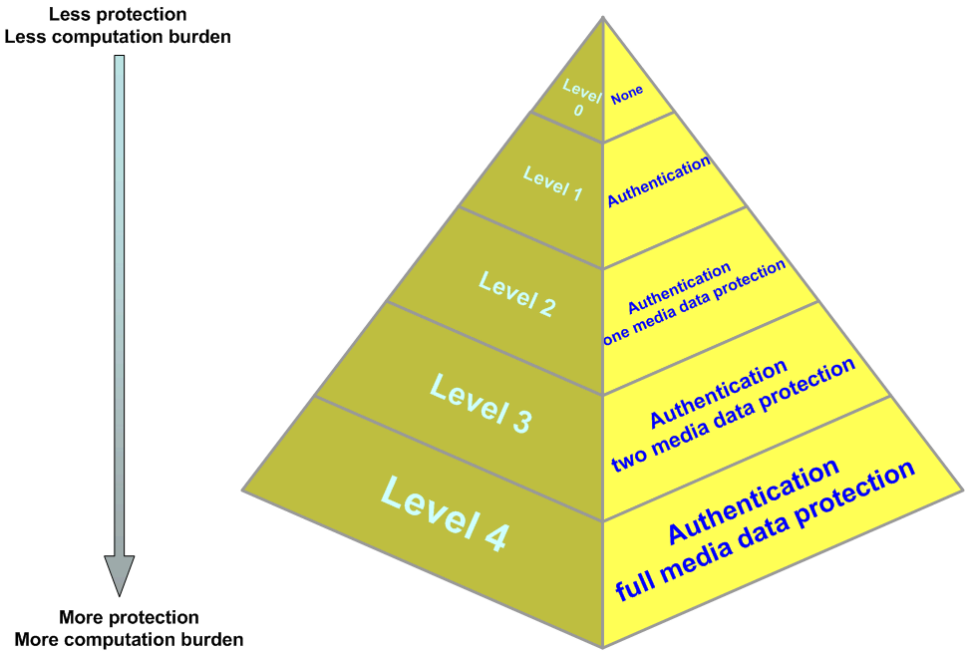


Figure 5.2: Security level pyramid

A conference usually contains three kinds of media data, i.e. video, audio, and shared data, e.g. whiteboard. At the beginning of a meeting, the initiator should determine whether the conference requires security or not. If true, users should be able to decide which kinds of media data will be protected in the conference. As a result, five security levels are distinguished (see Figure 5.2). The higher the security level the more protection is given to the conference. It is obvious that a fully protected conference poses the heaviest computational burden on the system. The five security levels are introduced in the following. Note that the signaling data are always protected except for level zero.

- *Level zero*: A level zero conference corresponds to a normal conference, where no special security function is applied.

- *Level one*: In a level one conference the joining of the group involves a mutual authentication, but the media data exchange is not further protected.

- *Level two:* In addition to the mutual authentication, a level two conference encrypts one media stream to be selected by the conference participants.

- *Level three:* A level three conference provides one more media stream protection than a level two conference. Users can choose two kinds of media data to be protected.

- *Level four:* Level four conferences are the most secure ones. All exchanged data are protected.

For security reasons, the system sets the level four as the default value. So all exchanged data are secured at the beginning of a secure conference. During the conference, users can adjust the security level based on their real security demands. For this, two different operation modes for managing the security policy were introduced: moderation and voting[2]. In the *moderation mode* the initiator is designated as moderator who solely decides all security demands. When the moderator leaves the conference, he/she can hand over the moderation right to one of the remaining members. In the *voting* mode all group members share the same right to decide about the security policy. The security policy used in the conference is determined by voting.

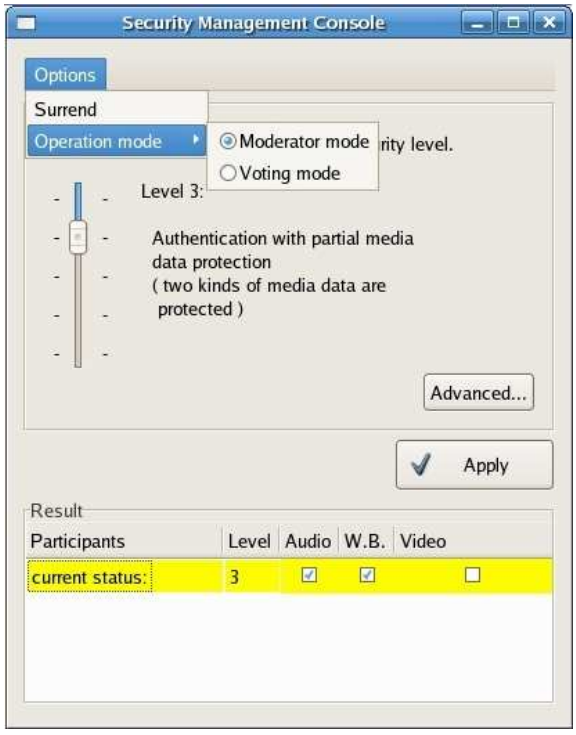


Figure 5.3: Security management console of BRAVIS

[2] The moderation and voting mode of the security policy module work in a similar mechanism with that of the floor control module. In principle they can be merged into the floor control module. At the moment, they are developed independent of the floor control module.

The graphic user interface (GUI) used for the security policy management is depicted in Figure 5.3. The GUI is mainly divided into two parts: upper part and lower part. The upper part is used for a participant to choose a set of security parameters that he/she consider appropriate for the meeting. He/she can vary the security level by dragging the slider bar and choose a set of crypto algorithms by clicking tab "Advanced". In addition, he/she can determine which operation mode should be applied during the meeting. The lower part of the GUI displays the current status of the security policy enforced in the meeting.

5.2.2 Authorization module

The authorization module is used to control the entrance into meetings and to govern the resource access. The former determines who has right to join a meeting. The latter ensures that a certain resource can be accessed only by the authorized users.

Entrance control

In BRAVIS the entrance into the meeting is by invitation. Each participant in the meeting can invite a new partner based on a social agreement with the other partners. No constraint is imposed on the callers for their calling activities, but an authorization function is applied to the invitee. Each invitee on its own decides to which incoming calls he/she will response. This is achieved by the use of an access control list (ACL) which is maintained by each participant. When a participant receives an invitation message, the required mutual authentication procedure is invoked. If this authentication is successful, the invitee has to check its ACL to examine whether the inviter has the right to call him/her. If true, it may accept this call. Otherwise the system can immediately reject the call without human intervention. Thus the ACL in BRAVIS actually acts as an automatic incoming calls filter so that malicious calls can be avoided.

Resource access

There are at least two mechanisms that can be used for the access control to resources (video, audio, and shared data (e.g. whiteboard)) in a conference. The first one is the all-or-nothing policy. The participants who have been allowed to enter into a meeting can access all resources in the meeting, whereas non-members cannot access any resources at all. This approach is appropriate for most conference scenarios in which participants interact in a natural way and no differentiation regarding the resource access is made among the participants. However, in some cases, a refined resource access mechanism which allows fine-grained control of access to individual resources is highly desired. For example, in a business meeting, maybe, only the chairman has the right to write on the whiteboard, while other participants can merely read the whiteboard. A number of approaches are available for the refined resources access control. The typical one is the role based access control

(RBAC) [181] in which the permission to access a resource is assigned according to the role of participants, and different roles have various dimensions for the resources access. Each participant is assigned a role when he/she is present in a conference and thereby can gain access to the corresponding resources that is granted to that role.

5.2.3 Group key management module

The group key management module is the key building block in the whole security architecture, since the security of all applied cryptographic protocols or algorithms merely rely on the privacy of the used key. The group key exchange protocols can be distinguished in centralized and decentralized (or distributed) protocols. The centralized approach uses a key server which supervises the group composition and generates a new group key if required. The centralized approach is less suited for peer-to-peer applications, since no extra server is used for this. The group members have to manage this task. Therefore we have to apply a distributed group key exchange protocol. Several protocols are available for this purpose such as CLIQUES [100], TGDH [101], the protocol proposed by Rodeh et al. [102], and others. However, they still possess shortages in respect of security and efficiency. To overcome the shortages of existing protocols we designed and implemented an efficient and secure decentralized group distribution protocol for BRAVIS, called VTKD (*virtual token based key distribution*) [103]. VTKD consists of two parts: a mutual authentication of the partners and a secure key renewal. The latter is triggered when the group composition changes, i.e. when members join or leave the group. The public key signatures based mutual authentication between the inviting group member and the invitee is invoked when a new member joins the group. This ensures that the group key is only delivered to an authenticated member, while the new member can be sure that the received key is in fact shared with the inviting parties. In the next chapter we will give a comprehensive introduction to the VTKD protocol regarding its principle, security features, and performance.

5.2.4 Data security module

The data security module is used to ensure the confidentiality and integrity of the data exchanged in the conference by using the group key agreed in the group key management module.

Data integrity

Standard data authentication schemes, such as HMAC [65], can be directly applied to signaling data and shared data to verify their integrity in transit. Any incidental transmission errors and malicious manipulations on the path can be detected using these schemes. A failed integrity check forces the receiver to ask for the sender to retransmit them until they

are transmitted correctly. Such procedures are applicable to the signaling data as well as to the shared data, since the signaling protocol GCP is a reliable protocol and the whiteboard or shared applications run on top of the reliable protocol TCP. The retransmission mechanism is an inherent feature for a reliable protocol. However, the real-time video/audio data are delivered over the unreliable protocol UDP where an error packet is not allowed to retransmit for strict real-time requirement reasons. Thus traditional crypto hash schemes are not well suited for the video/audio integrity verification because they cannot differentiate a transmission error from a content modification, and the packets are dropped in both cases. Recently, some research efforts have been devoted to addressing this challenging issue by proposing content authentication approaches for video/audio integrity such as feature extraction approaches [105], [106] and fragile watermarking approaches [107], [108]. Their objective is to identify whether the content of video/audio is altered or not, not to determine whether every bit in the video/audio data is modified or not. So the receiver can still play these video/audio containing errors as long as their content is not tampered by an attacker.

Data confidentiality

For the four kinds of data (video, audio, whiteboard, signaling), the participant can separately select different security algorithms for their protection. Standard encryption algorithms are used to process audio, whiteboard, and signaling data in real-time due to their small volume of data. To well meet the stringent QoS requirements of real-time applications a specific encryption algorithm is strongly demanded for video encryption due to the large amount of video data. For example, the bit rate of MPEG-2 video streams typically ranges between 4 and 9 Mbps [108]. For that reason, we developed a novel video encryption algorithm called *Puzzle* [109], [110] which is fast enough to meet real-time demands and provides a sufficient security meanwhile. The detailed introduction into *Puzzle* algorithm is left to the chapter 7.

Recently, the IETF released the standard SRTP (*secure real-time transport protocol*) [139] used for providing the integrity and confidentiality of RTP payload. The payload may contain video or audio data. There are at least two limitations in this standard. First it only specified the AES algorithm used for the encryption of video and audio data. How other algorithms can be added to the SRTP framework is not exactly specified. As discussed above, video data encryption usually needs specific algorithms rather than a standard algorithm (e.g. AES) to meet the stringent real-time requirements. Therefore the application of SRTP to video encryption may be not the best choice. Second it applies the standard HMAC function to ensure the integrity of the video and audio data. A single bit error in the transmission can cause the loss of a whole RTP packet due to the standard HMAC function, although this single bit error may not change the content of the video or audio. As a result, the quality of the video or audio is decreased perceivably. The resulting video and audio

quality may be even worse for a radio link, since its bit error rate (BER) is much higher than that of the wired link. To sum up, more studies are still needed to determine, whether or not the SRTP should be deployed in the BRAVIS system for video and audio protection.

5.2.5 Interactions between security modules

The functionalities of each security module have been introduced separately. This section illustrates how they harmoniously work together to provide the security services necessary for a secure conference.

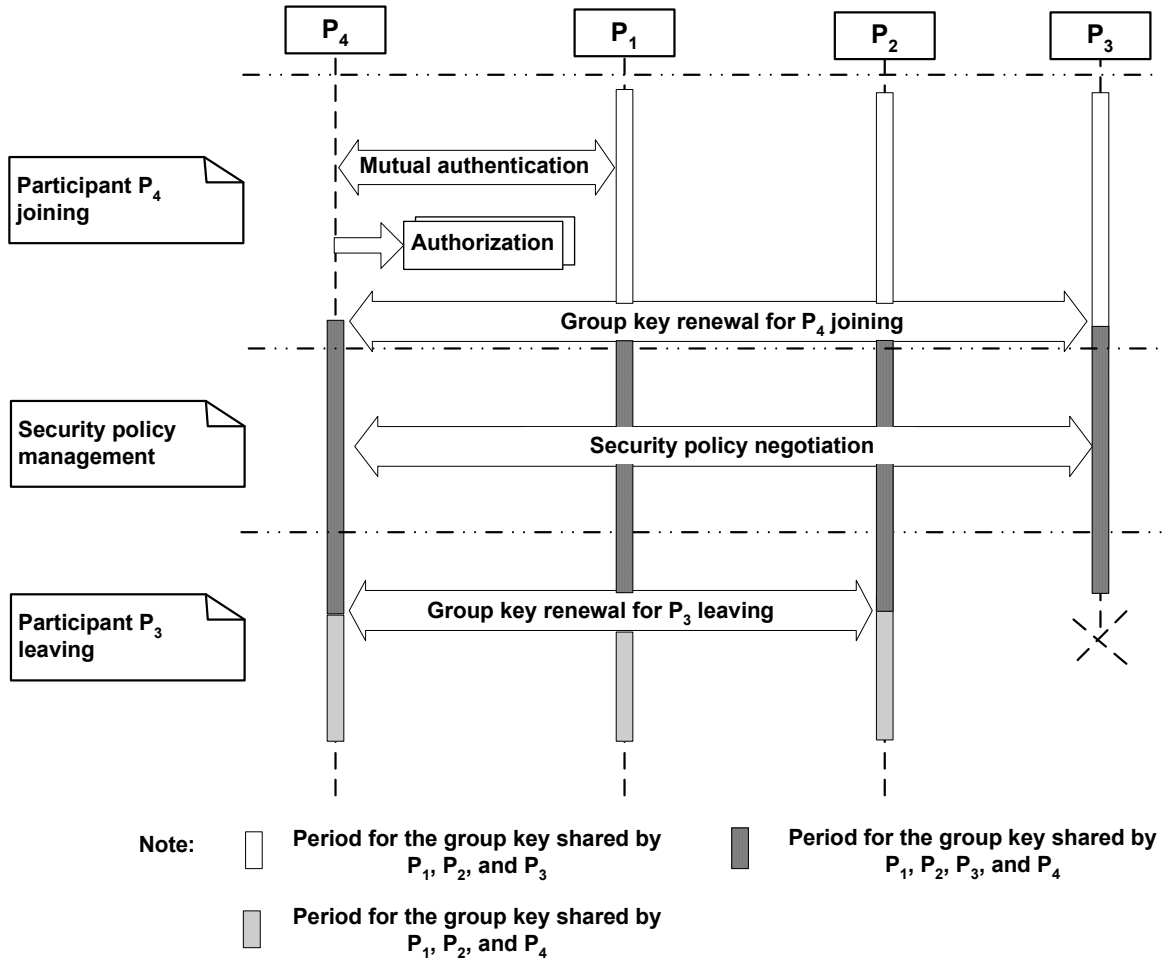


Figure 5.4: Interactions between security modules

The interactions between the security modules are elaborated through introducing the operation procedure of a secure conference. As shown in Figure 5.4, we assume that participants P_1, P_2, and P_3 have been in a secure conference. The participant P_1 is assumed to be the delegate of the group at this moment who initiates the group key management module

in response to the group composition change. The new participant P_4 is invited to join the conference by one of three members P_1, P_2, and P_3. Thereby the mutual authentication between the newcomer and the participant P_1 is invoked. After successfully authenticating participant P_1 the new participant P_4 learns who have been in the conference. He/she consults the authorization module to examine whether these members have the right to call him. If this is not true, participant P_4 simply refuses the call. Otherwise he accepts the call; the corresponding group key renewal procedure for his joining is triggered. Besides the distribution of the new group key, the group key renewal procedure is additionally used to deliver the current group security policy to the new member. So the members in the newly constructed group (P_1, P_2, P_3, and P_4) can securely communicate to each other under the same group key by obeying a unique security policy. During a secure group communication the members can negotiate a new security police by using the security policy module if they are not satisfied with the old one. Like the joining event, the leaving event can invoke the group key renewal operation as well. When the group key renewal is accomplished for the leaving of a participant, the remaining group members communicate using the new group key. As shown in Figure 5.4, participants P_1, P_2, and P_4 continue to securely talk to each other under the new group key after participant P_3 left. Whenever data are encrypted and hashed, the data security module will inquire the group key management module to identify whether the new group key is installed. If true, the data security module will apply the new key for data integrity and confidentiality. As shown in Figure 5.4, the data security model has received three different group keys due to the change of group composition.

It can be observed from the interactions between security modules that the group key management module is the most important one of them. Thus the first thing to realize the security architecture is to develop a secure and efficient group key management module. The next chapter focuses on this crucial issue.

Chapter

6

A Key Distribution Protocol for Small Peer Groups

The group key management protocol is the heart of the security architecture for group communication applications. It plays a decisive role to ensure group privacy and integrity. A compromise of the group key renders cryptographic algorithms used in a system completely meaningless. Designing a perfect group key management protocol always represents a challenging task. Group key management has to fulfill a set of security requirements to assure that only authorized group members can access to the group key. In addition, real-time settings such as video conference systems strongly require efficient protocols to be applied to better support services. Currently available protocols possess shortages in meeting these security and efficiency requirements demanded in real-time settings. In this chapter, we propose an efficient and secure protocol, called VTKD, which has lower rekeying delay than the existing protocols.

6.1 Requirements to group key management protocols

Small peer groups

Many emerging interactive and collaborative applications tend to apply the peer-to-peer paradigm to be independent of expensive infrastructures as they are, for instance, provided for audio and video conferences by the H.323 systems. Decentralized P2P systems better support spontaneity and mobility to set up meetings at varying locations or in ad hoc environments. This is especially advantageous for business communication over the Internet, but also other collaborative applications such as audio/video conferences, web conferences, and multiparty games. These applications usually need group privacy and data integrity. Decentralized solutions require appropriate mechanisms to protect the confidentiality of the communication. To assure confidentiality the partners have to agree upon a common secret key for encrypting their communication. While centralized collaborative systems provide practicable solutions for this, the development of efficient and secure key exchange procedure is still under research for decentralized systems.

Two different kinds of approaches are applied for this purpose: centralized and decentralized (or distributed) group key exchange protocols. The centralized approach uses a key server which supervises the group composition and generates a new group key if required. The centralized approach is readily to implement, but it is less suited for peer-to-peer applications. The key server may become a single point of failure and presents an attractive target for attacks. The distributed approach assigns the key management function to the group members.

Our intention is to design a simple key management protocol to efficiently support small dynamic peer group meetings. Small group peer-to-peer meetings are dominant in everyday life such as business talks, conferences, consultations, teleseminars, multiparty games etc. Interactive and collaborative meetings tend to be much smaller compared to the open multicast meetings set up via the Mbone [111].

Security and efficiency requirements

The group key management for small dynamic peer group meetings has to fulfill different requirements [112], [70]:

- *Key authentication:* Every group member has to assure that nobody outside the group acquires the group key. This requires a mutual authentication of the partners when joining the group to assure that the invited partner is the expected one, and vice versa that the invitee has certainty that he/she can trust the group.

- *Forward confidentiality:* Group members that leave earlier should not have access to any key generated later to decrypt data exchanged after their leaving.

- *Backward confidentiality*: Members joining later should not have access to any older key to decrypt data exchanged previously.

- *Collusion freedom:* Any subset of members that left the session should not be able to deduce the current key using former keys.

- *Perfect forward secrecy:* A compromised key can not lead to the disclosure of past keys.

- *Resistance to known key attacks:* The disclosure of past session keys cannot be used to compromise the current session key.

Besides security requirements stated above, efficiency is a crucial concern for the application of a group key management protocol in real-time settings. There are two primary reasons for this. On the one hand, real-time group communication applications like multi-party audio or video conferences make high efficiency demands (in a multi-party conference, for instance, the audio/ video streams of all participants have to be decompressed simultaneously). On the other hand, group communication is interfered during group key refreshment. This is because hosts are usually unable to update their group key synchronously in the asynchronous Internet [113], [114].

6.2 Related work

Group key management protocols can be generally classified into centralized and distributed protocols [115] depending on the fact whether the group key renewal is uniquely managed by a dedicated entity (e.g. key server) or collaboratively performed by the group members themselves. The former are designed for large group applications using one-to-many communication model such as video on demand service (VOD), satellite broadcast, and so on, where the group size may be more than one million members but there are relatively loose security requirements [116], [117]. The latter aim at interactive and collaborative applications using a many-to-many communication model, such as video conference, collaborative document sharing/editing, and so forth, which usually have small group size less than hundred members but rigorous security requirements [118], [119]. Distributed protocols are the focus of this introduction. Beforehand we give an overview to centralized protocols since many distributed protocols borrow many concepts from them, such as key tree. A Taxonomy of group key management protocols is illustrated in Figure 6.1.

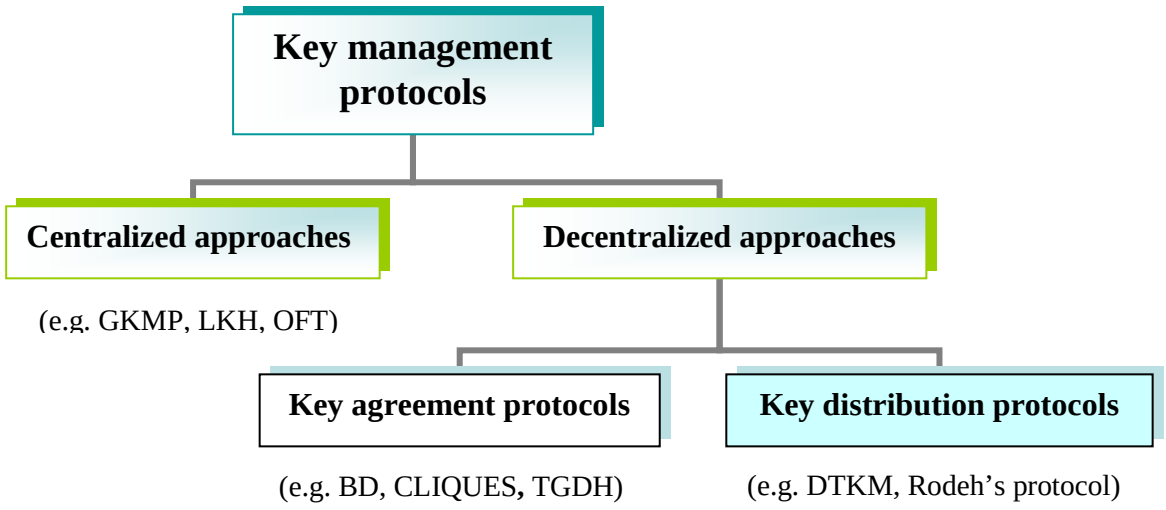


Figure 6.1: Taxonomy of group key management protocols

Centralized protocols

The *Group Key Management Protocol* (GKMP) is the simplest centralized approach used for the group key management [117]. The key server agrees upon a secret key with each group member. It delivers the new group key to each member encrypted with the corresponding secret key whenever required. This scheme is not efficient, because it requires *n* messages and *n* encryptions for a rekeying event where *n* is the group size. Wong et al. proposed the *Logical Key Hierarchy* (LKH) protocol [120] in which the key server maintains a key tree. The nodes of the tree are associated with intermediate keys, known as key encryption keys (KEKs) which are used for refreshing the group key and other KEKs. The leaves of the tree correspond to secret keys between the key server and each member. The root of the tree is the group key. Each member knows its leaf secret key and all KEKs from its leaf from the root. For each group key refreshment operation, the key server needs only to change the keys known to the left member on the path from its leaf to the root. As shown in Figure 6.2, the key server merely updates $K_{1\text{-}8}$, $K_{5\text{-}8}$, $K_{7\text{-}8}$ when member P_8 left the group. LKH reduces the number of rekeying messages and the number of encryption from *n* in GKMP to $2log_2\ n$. One-way function tree (OFT) [121] scheme is an improvement to the LKH approach, which further reduces the rekeying cost of the LKH by half. It is worth to mention that the rekeying efficiency of LKH and OFT mainly relies on a balanced key tree. After many rekeying operations the key tree may become imbalanced. To keep the efficiency of LKH and OFT it is necessary to rebalance the key tree [122].

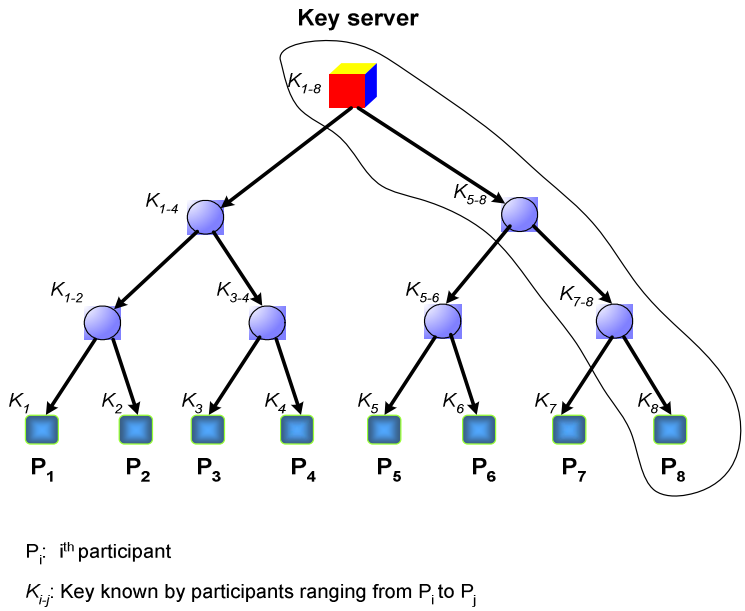


Figure 6.2: LKH tree

Decentralized protocols

Distributed group key management protocols can be divided into two categories: group key agreement and group key distribution protocols [123]. *Group key agreement protocols* are based on the two-party Diffie-Hellman key exchange protocol [124]. Their basic idea is that each group member has to contribute a share to generate the group key. When the group membership changes, a group member is selected to compute new intermediate keys and distribute them to the group. Based on these intermediate keys and its own share each group member can independently compute the group key. Examples of such protocols are BD [118], CLIQUES [100], and TGDH [101]. One of the first proposals was the approach of Burmester and Desmedt [118]. They proposed a computation-efficient protocol to refresh the group key for any membership change. However, it requires a high communication overhead of 2*n* broadcast messages. CLIQUES [100], developed by Steiner, Tsudik, and Waidner, is a natural extension of the Diffie-Hellman protocol for dynamic peer groups. Each member adds its share to the intermediate value generated by the predecessor and passes the new value to the successor. The last group member in the chain adds its share to all intermediate values and multicasts them to the group members. Each group member determines the group key using the respective intermediate value and its share. The number of cost-expensive exponentiation operations and rekeying messages increases linearly with the group size. The main weakness of CLIQUES is the high computational overhead that it imposes on the last member in the chain.

Kim, Perrig, and Tsudik proposed a *Tree based Group Diffie-Hellman* (TGDH) protocol [101], which is built by combining the two-party DH protocol with key trees concept of LKH. The key tree is arranged as follows. Each node is associated with a DH private key and the corresponding DH public key (called blinded key in the original paper). Each leaf represents a group member, who has own private key and the associated public key. Every member holds all private keys on the path from its leaf to the root as well as all public keys on the key tree. Therefore, the private key held by the root node is known by all members and is used as the group key. The basic idea of TGDH is that every member can compute the private keys along its key path to the root node based on his own private key, by recursively using the two-party DH protocol. For each group key renewal, one group member, the so-called *sponsor*, generates new intermediate public keys and distributes them to the group. Each member computes the new group key using these intermediate public keys and its own share. The key tree reduces the communication and computation overhead for refreshing the group key compared to CLIQUES. The amount of rekeying messages and exponentiation operations in TGDH is reduced from $O(n)$ in CLIQUES to $O(\log_2 n)$. TGDH proved to be the most efficient protocol among the above mentioned group key agreement protocols related to computational and communication overhead [123].

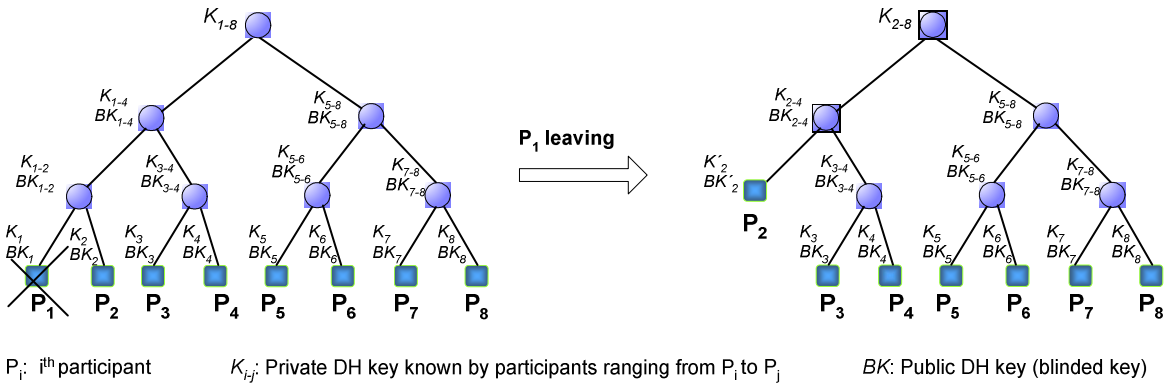


Figure 6.3: Group key renewal of TGDH for a leaving event

We take the following example to illustrate how participants agree upon a new key again when the group composition changes. Figure 6.3 depicts the group key renewal of TGDH for a leaving event in a group of 8 members. We assume that participant P_1 leaves the group and participant P_2 becomes the sponsor. After updating the tree, the sponsor generates a new private DH key K_2`, recalculates the private DH keys $K_{2\text{-}4}$, $K_{2\text{-}8}$ as well as public DH keys $BK_2$`, $BK_{2\text{-}4}$. Then it broadcasts the public DH keys BK_2`, $BK_{2\text{-}4}$ to the group. Participants P_5, P_6, P_7, and P_8 can compute the new group key $K_{2\text{-}8}$ upon receiving the public DH key $BK_{2\text{-}4}$. In a similar manner, participants P_3 and P_4 can sequentially compute the private DH key $K_{2\text{-}4}$ and new group key $K_{2\text{-}8}$ after receiving the public DH key BK_2`.

In contrast to group key agreement protocols the *group key distribution* approaches dynamically select one group member to generate and distribute the new group key. Dondeti et al. proposed a distributed tree-based key management scheme (DTKM) for secure many-to-many group communication [125]. This approach modified the centralized OFT (*One-way Function Tree*) approach of [121] to a decentralized one in which one group member is dynamically selected to refresh the group key. This approach, however, has an expensive communication overhead. It demands $log_2 n$ rounds to complete the group key refreshment for a join or leave event. An alternative distributed key tree approach was suggested by Rodeh et al. [102]. It is an extension of the centralized LKH protocol. In the Rodeh protocol all keys used by the group key management are arranged in a key tree. The leaves of the tree correspond to group members. The left-most leaf is defined as the *tree leader*. For each group key renewal, the tree leader directly generates the new group key and sends it to the subtree leaders via secure channels which are generated by exchanging public DH keys between the tree leader and subtree leaders. The subtree leaders continue sending the new group key to their respective subtree members. As shown in Figure 6.4, given a group of 8 members, we assume that participant P_1 leaves the group. In response to the leaving of P_1, the tree leader P_2 generates the new group key $K_{2\text{-}8}$, and securely sends it to the subtree

leaders P_3 and P_5 first. Then they deliver the new group key to their subtree members individually in the second communication round, i.e. ($P_3 \rightarrow P_4$) and ($P_5 \rightarrow P_6$, $P_5 \rightarrow P_7$, $P_5 \rightarrow P_8$). Compared to DTKM, this reduces the communication cost to two rounds for a key refreshment event. It is, however, not resistant to *known key attacks*, since the tree leader encrypts the new group key with the old group key and multicasts it to group members for a join event.

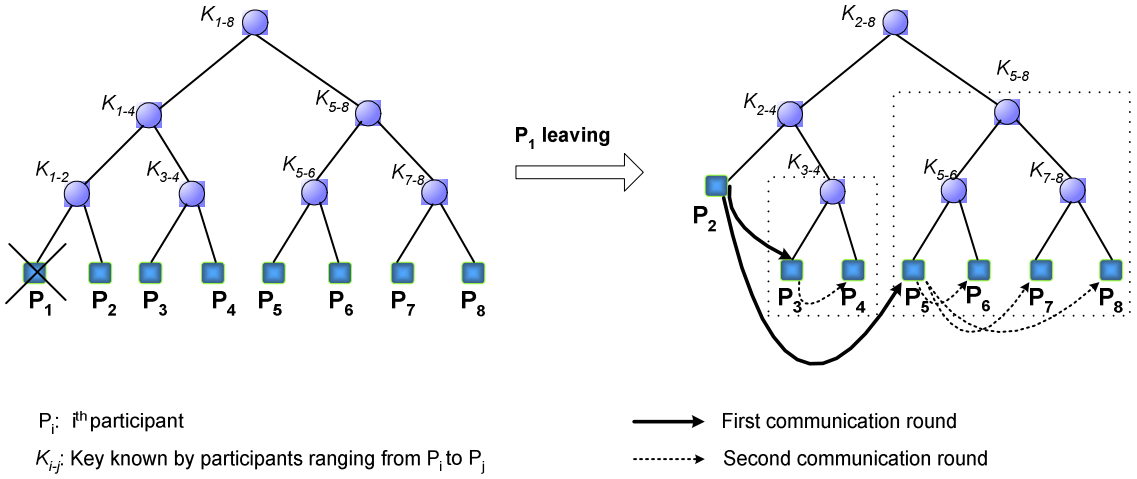


Figure 6.4: Group key renewal of Rodeh's algorithm for a leaving case

The above mentioned distributed key agreement and distribution protocols do not consider *key authentication* as part of the key management protocol. Distributed key management protocols seldom support this property. Ateniese et al. presented the protocol SA-GDH (*Secure Authenticated Group Diffie-Hellman*) [126], a derivate of CLIQUES, which includes a key authentication feature. However, several possible attacks on this protocol have been reported meanwhile [127].

A comparison of both approaches shows that distributed key agreement and distribution protocols do not differ much related to the group key renewal procedure. The essential difference between them is that in key agreement protocols one selected group member generates the intermediate keys rather than the group key, whereas key distribution protocols select a certain group member to directly generate the group key. Group agreement protocols, however, require more complex data structures for the key calculation and are more expensive regarding the computational overhead than group distribution protocols [128]. In view of the efficiency demands of real-time group communication we prioritize a decentralized distribution protocol. As discussed above several decentralized distribution protocols have been proposed, but they still possess shortages in the respect of security and of efficiency as mentioned above.

6.3 Principle

To overcome the shortages of existing protocols, we have designed two efficient and secure decentralized group distribution protocols, called TKD (*Token based Key Distribution*) [133] and VTKD (*Virtual Token based Key Distribution*) [103] respectively. The TKD protocol is an early version of the VTKD protocol. They work in compliance with the similar principle. Before introducing this principle, we first describe the system architecture assumed.

6.3.1 System architecture

To explain the integration of TKD/VTKD into a system architecture as well as the interaction with other protocols we use our multi-party video conference system BRAVIS [17] as example. TKD/VTKD can be similarly integrated in other systems.

The architecture assumed for TKD/VTKD consists of 3 layers: an application layer, a security layer, and a group communication layer (see Figure 6.5). The *application layer* needs not to be specified in detail here. It contains the application-specific functions. In BRAVIS these are the conference control modules such as the QoS management and the floor control as well as the media transfer modules such as the video and audio manager. The related descriptions can be found in the chapter 2 and chapter 5.

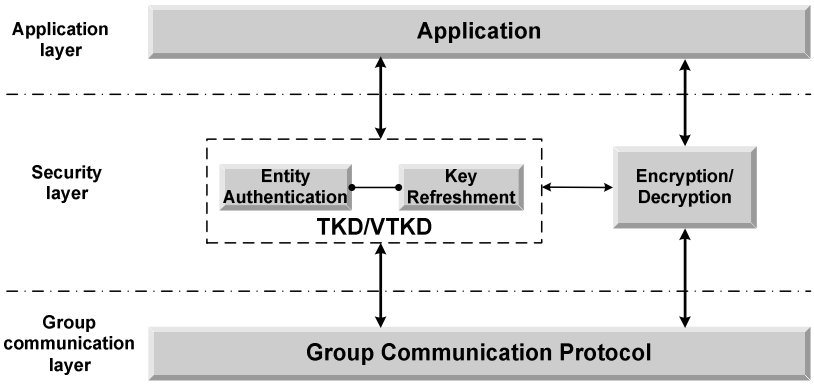


Figure 6.5: System architecture

The *security layer* contains the encryption/decryption module for data and media exchange and the TKD/VTKD protocol which is the focus of the consideration here. The key renewal is triggered when the group composition changes, i.e. when a new partner joins the group, a participant leaves the group or a participant's host crashes. The joining of the group always involves a mutual authentication to assure that both sides can trust each other. The security layer is closely connected with the group communication layer which assures the consistency of the group data.

In collaborative applications the *group communication layer* constitutes the basis for a reliable operation of collaborative peer-to-peer applications. It has to update the group management data in the peers and to ensure their consistency so that all peers have the same view on the actual group state and can uniquely decide all group related issues by themselves, e.g. QoS parameter settings or floor assignment. In closed groups it also has to ensure the closeness of the session. To achieve this, a group communication protocol is required that provides virtual synchrony [47]. This assures that no data are lost, that data are delivered in the order as they are sent, and finally that all peers are updated equally. There are several protocols which meet these requirements like RMP [129], the Totem Protocol [130], Ensemble [131], Spread [132], and GCP [45], [46]. In BRAVIS we use the latter protocol. Decentralized key management protocols heavily depend on the virtual synchrony property of the group communication protocol for refreshing the group key [101], [102], [123]. If this property is not provided, members may have a different view on the group membership when key renewal is required. This leads to confusion in the renewal process, since more than one member may be selected to generate the group or intermediate keys, respectively. Therefore, we assume like other decentralized key management protocols that a group communication protocol with virtual synchrony is applied in the communication layer.

The group management is contained in this layer. It supervises the group composition and indicates all changes to the group members. The group management also triggers the key refreshment when members join or leave a session.

The group management also executes the invitation procedure. In closed group meetings the identities of all participants are managed in a directory infrastructure deployed in an enterprise or domain, i.e. all participants belong to the same trust infrastructure or namespace. Thus identity forgery, i.e. a Sybil attack [44], is not an issue for such group settings. The group is set up by an initiator who invites the partners. Later, further partners can join the group if desired. The decision to invite new partners is based on social agreement of all partners.

6.3.2 Mechanism of TKD/VTKD

This subsection is organized as follows. First the DH principle used as the basis of VTKD is briefly reviewed. Secondly the associated security assumptions for VTKD are made. Then the mechanisms of the TKD and VTKD protocol are separately introduced. More details are given to the VTKD protocol.

Diffie-Hellman (DH) key exchange protocol [62]

The DH protocol enables two partners to agree upon a shared secrete key by exchanging their public DH keys (values). The security of DH protocol depends on the difficulty to computing discrete logarithms for large numbers. All arithmetic operations of DH protocol are performed on the cyclic group of prime order p with generator g. Provided that user Alice and user Bob want to set up a secure communication, Alice selects a random number r_a as the secret DH key and generates the corresponding public DH key $g^{r_a} \bmod p$, similarly Bob possesses a secret DH key r_b and the corresponding public DH key $g^{r_b} \bmod p$. After exchanging their public DH keys, Alice and Bob can individually compute a shared secret key SK as follows:

$$SK = (g^{r_a})^{r_b} \bmod p = (g^{r_b})^{r_a} \bmod p = g^{r_a r_b} \bmod p$$

For simplicity, we omit the term "mod p" when expressing public DH values in the later sections of this thesis.

Security Assumptions

TKD/VTKD is a decentralized group key distribution protocol which is based on the Diffie-Hellman (DH) key exchange principle. There is no central group key authority. In contrast to the key exchange between two partners, in the distributed approach each group member calculates a secrete key with each partner using the Diffie-Hellmann principle. This key is called shared DH-secret in the sequel. They are stored at each member and used for the group key distribution. Concerning the group members it is assumed that all members have equal rights and possess the same trust, i.e. each member may authenticate new members and trigger the group key refreshment. We further assume that an authenticated member in a closed group meeting is trustworthy, i.e. he/she does not actively attempt to disturb the system and to disclose the group key to non-members. No assumptions are made on the trustworthiness of partners after leaving. These assumptions correspond to practical security demands. The decentralized group key protocols mentioned in Section 6.2 rely on similar assumptions.

Mechanism of TKD

TKD is a token based protocol. The group members form a logical ring based on the group membership list generated in the group communication layer. The token determines the group member that generates a new group key and initiates the key distribution procedure. The group key is renewed whenever the group composition changes (join, leave, and failure of peers). The token principle was chosen to select the member responsible for the group renewal process in this dynamic group configuration. For smaller groups, as assumed here, the token approach is efficient enough. The token holder is also the group member who authenticates the joining partners. The initiator of the group creates the first token. After renewing the group key the token holder hands the token over to the next group member in

the ring. The token shift is part of the rekeying message which is multicast to the whole group. Thus each group member knows the current token holder at any time. The reliable delivering of the rekeying message is guaranteed by the underlying group communication layer as discussed above.

Mechanism of VTKD

The basic idea of VTKD is the same as TKD, i.e. using a token to determine the partner responsible for the key generation and distribution procedure. The only difference between two schemes is related to the token management. VTKD does not use a physical token designed in TKD, which rotates among the group members, but a virtual one. This means that the position of the token holder is newly determined for each key distribution. Thus all problems associated with the explicit token delivery are avoided such as token loss and duplication. The token position is computed based on the *group member list* (see Figure 6.4). The group member list is organized as follows. The entry determines the position of the member. The members enter into the list in the order they join the group. When a member leaves the list entries are shifted. The new token position *PT* is determined as follows:

$$PT = VK \bmod n \tag{6.1}$$

where *VK* denotes the version of the group key and *n* the current number of group members. *VK* is increased by 1 each time the group key is renewed. The function of *VK* in our protocol is two-fold. Besides determining the token position it is also used to counter replay attacks. The virtual synchronization property provided by the underlying group communication protocol ensures that each group member knows the current group size and key version in a consistent view. Thus each group member can clearly determine the position of the virtual token in the group member list.

In VTKD each group member executes the same security functions. When the group composition changes the token holder refreshes the group key. It creates a separate temporal secure channel to every group member to deliver the new group key. To set up the channels it uses the shared DH secrets and a newly generated nonce. A short example is supposed to explain the principle (see Figure 6.6). We assume a group of four members (P_1, P_2, P_3 and P_4). P_1 is holding the token. Each member knows its shared secrets with the other members. For example, P_1 stores $g^{r_1r_2}$, $g^{r_1r_3}$, and $g^{r_1r_4}$, P_2 accordingly $g^{r_2r_1}$, $g^{r_2r_3}$, and $g^{r_2r_4}$. When the group composition changes P_1 sets up the separate temporary secure channels K_{12}, $K_{13,}$ and K_{14} with P_2, P_3, and P_4 using the shared secrets $g^{r_1r_2}$, $g^{r_1r_3}$, $g^{r_1r_4}$, and the nonce N_1. It can now securely deliver the new group key to P_2, P_3, and P_4 via these temporal secret channels.

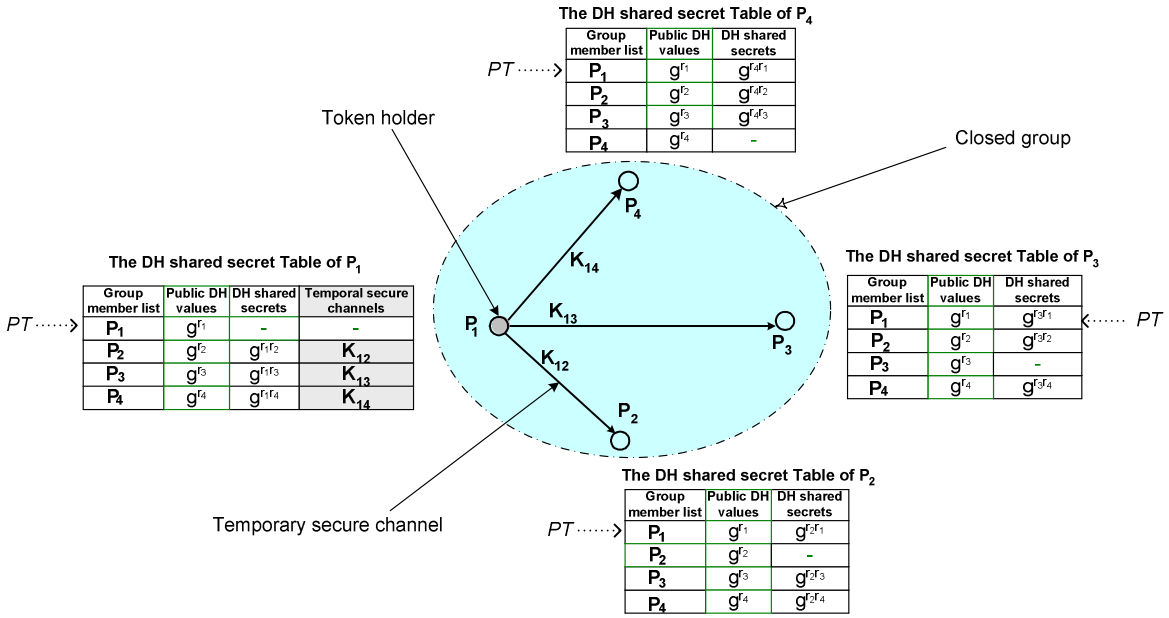


Figure 6.6: Key exchange in VTKD using temporal secure channels

The secure channels K_{ij} from the token holder P_i to the group members are established by using two temporary shared keys: the encryption key $K_{ij\text{-}e}$ for encrypting the exchanged messages and the authentication key $K_{ij\text{-}a}$ used for message authentication. For key generation, a pseudorandom function is applied which hashes a message *m* using a key *k*. We use HMAC(*k*,*m*) [65] here. The keys are generated as follows:

$$K_{ij-e} = HMAC(g^{r_i r_j}, g^{r_i r_j} | N_i | ID_i | ID_j | 0) \tag{6.2}$$

$$K_{ij-a} = HMAC(g^{r_i r_j}, g^{r_i r_j} | N_i | ID_i | ID_j | 1) \qquad (j=1, 2,...n \text{ and } j \neq i) \tag{6.3}$$

where N_i denotes a nonce and ID_i the token holder's identity which is contained in the re-keying message sent by the token holder (see Section 6.4). ID_j is the group members' identity and $g^{r_i r_j}$ the secret key stored at both sides. The symbol "|" means concatenation.

The obvious precondition of VTKD is that each member can store the shared DH secrets with the other group members at any time no matter how the composition of the group changes. This condition is easy to fulfill. When a member leaves the session the remaining members simply delete the respective secret in their DH secret tables. In the reverse case it is more difficult, because the joining member as well as the group members do not know the public DH values of the opposite side. This problem is solved as follows. While the token holder and the invitee mutually authenticate the token holder sends all public DH values of the group members to the new partner. The token holder vice versa gets the public DH value of the new member and forwards it to the group. Each group member then com-

putes the shared secret related with the new member. Now each member knows again all shared secrets with the other members, i.e. each group member can refresh the group key when it becomes the token holder and the group membership changes.

6.4 Protocol procedures

In the rest of the chapter we concentrate only on the VTKD protocol since the VTKD and TKD protocol share the similar principle. The essential procedures of the VTKD protocol are given in the sequel.

6.4.1 Join procedure

The join procedure consists of two steps: (1) the authentication phase in which the invitee and the token holder mutually authenticate and (2) the proper join phase with the group key refreshment (see Figure 6.7). Five messages or rounds, respectively, are needed for the join procedure: four rounds for authentication and one for the key refreshment.

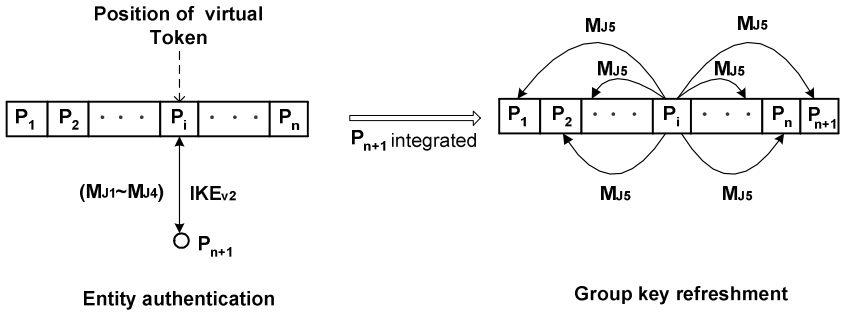


Figure 6.7: Join procedure

Authentication

For mutual authentication between the token holder and the invitee, the newly proposed internet draft standard IKEv2 [134] has been deployed for its increased efficiency, security, flexibility, and robustness compared to its predecessor IKE [96]. It can be expected that it will be widely used for authenticated key exchange in the Internet. To adapt it to the group communication scenario we had to change some message components.

IKEv2 supports two kinds of authentication: public key signatures and pre-shared secrets. We use public key signatures here which are more appropriate for peer-to-peer systems than the pre-shared secret authentication which better supports client/server applications. Moreover, as proved in [135], [136], the signature mode of IKEv2 is a secure authenticated key exchange protocol which thwarts those commonly appearing attacks on authentication

protocols such as man-in-the-middle and reflection attacks. The successful authentication of public key signatures mainly depends on the authenticity of the public keys. The certificate is the mostly accepted approach used for this purpose. Here we assume that a public key infrastructure (PKI) is deployed.

We now consider the case that P_{n+1} is invited to join a group of n participants: $P_1 \ldots P_n$ (see Figure 6.7). P_i is supposed to be the current token holder selected according to formula (6.1). To accomplish the mutual authentication between P_i and P_{n+1} the following four messages have to be exchanged.

$$M_{J1}(P_i \rightarrow P_{n+1}): \; HDR, g^{a_i}, SA_i, NA_i$$
$$M_{J2}(P_{n+1} \rightarrow P_i): \; HDR, g^{a_{n+1}}, SA_{n+1}, NA_{n+1}$$
$$M_{J3}(P_i \rightarrow P_{n+1}): \; HDR, SK\{ID_i, CERT_i, SIG_i, ID_1, ID_2 \ldots ID_n, g^{r_1}, g^{r_2} \ldots g^{r_n}\}$$
$$M_{J4}(P_{n+1} \rightarrow P_i): \; HDR, SK`\{ID_{n+1}, CERT_{n+1}, SIG_{n+1}, g^{r_{n+1}}\}$$

HDR in the four messages denotes the message header which contains token holder's SPI_i (*Security Parameter Index*), and the invitee's SPI_{n+1}. *SPI* is a value chosen by a user to identify a unique IKE security association. It is also used as a cookie to provide a limited protection from denial of service attacks.

The messages M_{J1} and M_{J2} have two functions: (1) to negotiate the security association *SA* between the token holder and the invitee, which specifies the cryptography parameters used for the messages M_{J3} and M_{J4}, and (2) to exchange their public DH values g^{a_i} and $g^{a_{n+1}}$ as well as a nonce *NA* to generate the shared keys *SK* and *SK`*, which are used to protect the subsequent messages M_{J3} and M_{J4}. Note that for a maximum security the public DH values g^{a_i} and $g^{a_{n+1}}$ used to construct the shared key between the token holder and invitee in the authentication phase is distinct from the public DH values g^{r_i} and $g^{r_{n+1}}$ used for temporal secure channels in the rekeying phase. Consequently, the shared keys *SK* and *SK`* are significantly different from the temporal secure keys $K_{ij}$ generated in the rekeying phase. To avoid a reflection attack, separate session key *SK* and *SK`* are used for each direction [134]. *SK* (and also *SK`*) consists of the encryption key SK_e and the authentication key SK_a. In general, it is required that distinct keys are used for encryption and authentication, respectively, so that an interconnection between the different mechanisms is avoided [70]. In other words, a compromised encryption key caused by a weak encryption algorithm should not allow compromising the authentication key and vice versa. The shared keys *SK* and *SK`* are computed as follows [134]. First a key material called *SKEYSEED* is calculated by using a random number generation function *prf* which outputs a pseudo-random stream. Its inputs are the nonce exchanged in messages M_{J1} and M_{J2}, and the DH shared secret agreed on this exchange. Second *SKEYSEED* is further used to generate an auxiliary key SK_d and

shared keys (SK_a, SK_a`, $SK_e$ and $SK_e$`) for protecting messages M_{J3} and M_{J4}. These computation procedures are expressed as follows:

$$SKEYSEED = prf(NA_i | NA_{n+1}, g^{a_i a_{n+1}})$$
$$SK_d = prf(SKEYSEED, g^{a_i a_{n+1}} | SPI_i | SPI_{n+1} | 0x01)$$
$$SK_a = prf(SKEYSEED, SK_d | g^{a_i a_{n+1}} | SPI_i | SPI_{n+1} | 0x02)$$
$$SK_a` = prf(SKEYSEED, SK_a | g^{a_i a_{n+1}} | SPI_i | SPI_{n+1} | 0x03)$$
$$SK_e = prf(SKEYSEED, SK_a` | g^{a_i a_{n+1}} | SPI_i | SPI_{n+1} | 0x04)$$
$$SK_e` = prf(SKEYSEED, SK_e | g^{a_i a_{n+1}} | SPI_i | SPI_{n+1} | 0x05)$$

In the original IKEv2 protocol M_{J3} and M_{J4} have two objectives. First the token holder and invitee mutually authenticate by verifying the signature *SIG* of the partner. Each peer generates the signature by signing the concatenation of its own first message (including its public DH value g^a), the partners' nonce and the value of *prf(SK,ID)* with its own private key. The token holder's signature SIG_i and the invitee's signature SIG_{n+1} are illustrated as follows:

$$SIG_i = PRK_i\{H(M_{J1} | NA_{n+1} | prf(SK_a, ID_i))\}$$
$$SIG_{n+1} = PRK_{n+1}\{H(M_{J2} | NA_i | prf(SK_a`, ID_{n+1}))\}$$

Note: H( ) means hash function
PRK_i, PRK_{n+1} token holder's and invitee's private key, respectively

Thus the signature verification of the partner not only authenticates the partner but also rules out any possibility of man-in-the-middle attack during the exchange of messages M_{J1} and M_{J2}, since an attacker in the middle cannot forge signatures without token holder's and invitee's private key used for signature, respectively. Secondly they agree upon a security association by exchanging their respective proposed security association which is used to protect the communication between these two participants. Since the agreed security association is only available for a two-party communication, this objective of IKEv2 is not preserved in VTKD which aims at group communication. Instead M_{J3} and M_{J4} are used to exchange information between the token holder and the invitee. The token holder delivers the group information to the new member in message M_{J3} including all members' identities (ID_1, ID_2,…, ID_n) and the respective public DH values (g^{r_1}, g^{r_2},… g^{r_n}). The invitee P_{n+1} returns its identity ID_{n+1} and its public DH value g^{rn+1} with message M_{J4}.

If the token holder fails to authenticate the invitee it notifies the group about this with message M_{Jf}:

$$M_{Jf}(P_i \rightarrow P_1, P_2 ... P_n): HDR, GK_{old}\{ID_{n+1}\}.$$

When receiving M_{Jf}, each member knows that the new member failed to join the group. All members keep the group key unchanged.

Group key refreshment

After successfully authenticating the invitee the token holder P_i renews the group key. The new key GK_{new} is randomly generated and thus independent of previously used keys. The token holder sends the new key with the multicast message M_{J5} to the extended group. For the exchange of M_{J5}, the secure temporary channels described in Section 6.3.2 are used. Message M_{J5} has the following format:

$$M_{J5}(P_i \rightarrow P_1, P_2 ... P_{n+1}): HDR, GK_{old}\{ID_i, N_i\}, K_{i1}\{VK, GK_{new}\} \\ ,..., K_{in}\{VK, GK_{new}\}, SK\{GK_{new}, VK, GSA, ID_i\}, GK_{new}\{g^{r_{n+1}}, ID_{n+1}\}$$

The message M_{J5} consists of four parts which serve different purposes. The first part of message M_{J5} contains the token holder's identity ID_i and a nonce N_i used to construct the temporal secure channels (see formula (6.2) and (6.3)) between the token holder and the other members. This part is encrypted with the old group key GK_{old}. The second part of message M_{J5} contains the new group key GK_{new} and the group key version VK. These data are separately encrypted for each group member using its temporal channel keys K_{ij} (j=1, 2,…n and j≠i) between the token holder P_i and the other group members. The third part of message M_{J5} for the new member contains the new group key GK_{new}, its version, the group key association GSA, which specifies the security policy currently deployed for group communication, and the token holders' identity. These data are encrypted with the shared key SK determined during the authentication phase (see above). The fourth part contains the new members' identity ID_{n+1}, and its public DH value $g^{r_{n+1}}$. This part is protected by the new group key GK_{new}.

After having received M_{J5} the old group members can decrypt ID_i and N_i with the old group key GK_{old}. They can determine the channel key according to formula (6.2) and (6.3) and decrypt the new group key GK_{new}. The new group member decrypts the new group key using the shared key SK. Now all group members including the new participant P_{n+1} possess the new group key, the public DH values of the other members as well as their shared secrets. They are all in the same state so that the new token holder can refresh the group key when required.

6.4.2 Leave procedure

When a participant leaves the group the underlying group communication protocol informs the remaining members about the leaving. Accordingly each group member updates its group member list. The group members determine the new token holder's position according to formula (6.1). The token holder starts the key refreshment procedure. Figure 6.8 shows an example.

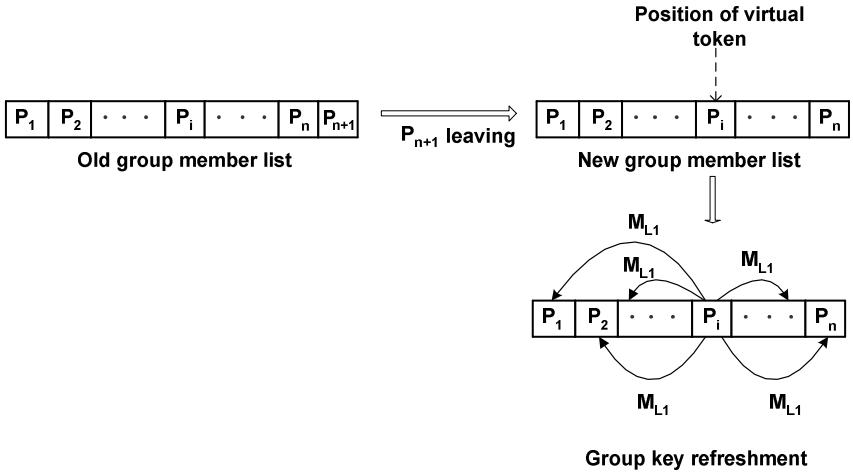


Figure 6.8: Leave procedure

We assume that participant P_{n+1} is leaving a group of n+1 members $P_1 \ldots P_{n+1}$. The token holder P_i generates a new group key GK_{new} and multicasts it in the leaving message M_{L1} to the remaining group members. M_{L1} has a similar structure like the join message M_{J5}:

$$M_{L1}(P_i \rightarrow P_1,P_2 \ldots P_n):HDR,GK_{old}\{ID_i,N_i\},K_{i1}\{VK,GK_{new}\},\ldots,K_{in}\{VK,GK_{new}\}$$

It first encrypts the identity of the token holder ID_i together with a newly generated nonce N_i with the old group key. The new key GK_{new} and the actual key version VK are encrypted with the temporary channel keys for each remaining group member. These channel keys are derived according to formula (6.2) and (6.3). The leaving participant cannot obtain possession of GK_{new}, because it cannot reconstruct any of the temporary secure channels K_{i1}, $K_{i2} \ldots K_{in}$ without knowing the shared secrets $g^{r_i r_1}$, $g^{r_i r_2}, \ldots g^{r_i r_m}$ between the token holder P_i and the remaining members P_1, $P_2, \ldots P_j, \ldots P_n$ ($j \neq i$). When receiving message M_{L1} the remaining group members can decrypt the new group key in the same way as described above for message M_{J5}. The group key refreshment is completed.

6.4.3 Failure of a participant

In VTKD the position of the virtual token is at any time known to the current group members. When changing of the group composition the new position can be determined exactly. As special case the crash of a host including the token holder must be considered. This is detected by the underlying group communication protocol and indicated to the security layer, i.e. the failure of a participant essentially coincides with the leave event. The security layer then invokes the leave procedure as described above.

6.5 Security analysis

We now analyze how VTKD fulfills the security requirements discussed in Section 6.1. Furthermore, we compare the security properties with that of TGDH and Rodeh's protocol which are considered the most efficient group key agreement and distribution protocols, respectively.

Key authentication is assured in VTKD, because each potential participant of a meeting must be authenticated by a group member. Only if this authentication is successful the invitee can join the group and obtain the group key. Vice versa, the new member authenticates the group information of message M_{J3} (their identities and public DH values) to convince itself that the received group key is of the expected group. The key refreshment procedure assures that only active group members can obtain the new group key by using separate temporary secure channels between the token holder and the other members.

Forward confidentiality is guaranteed by the leave procedure described in Section 6.4.2. The leaving member cannot access the new group key which is distributed to the remaining group members using the temporary secure channels between them and the token holder. These channels rely on the shared secrets between the token holder and the remaining members which are not accessible for the leaving participant.

Backward confidentiality is achieved by not delivering the old group key to the joining member as shown in Section 6.4.1. Those parts of message M_{J5} that can be decrypted by the new member do not contain the old key.

Collusion freedom is ensured, because each time the group composition changes the token holder randomly generates a new group key which is not associated with formerly used keys. For that reason, former participants are not able to deduce the current group key based on any subset of expired session keys.

Perfect forward secrecy means that either a compromised long-term credential will not result in compromising of expired session keys or an active attack on the system (e.g. a hacked host) will not reveal past session keys [137]. Perfect forward secrecy of the first case is trivially achieved, since the long-term credentials in VTKD (e.g. the private key used for signature during the authentication phase) are never used to encrypt any group keys

The second case is relevant, when a meeting consists of several sessions. A singular meeting is devoted to a particular activity that does not have any previous sessions. In a confidential meeting with several sessions various session keys may be used. For example, we

assume a confidential meeting of four sessions whereby the group key should be refreshed for each session. Each session is characterized by its group key and associated keying materials as shown in the following table.

Key materials	Session1	Session2	Session3	Session4
Group key (GK)	GK_1	GK_2	GK_3	GK_4
Temporary shared key between member P_i and member P_j (K_{ij})	K_{ij1}	K_{ij2}	K_{ij3}	K_{ij4}
Nonce (N_i)	N_{i1}	N_{i2}	N_{i3}	N_{i4}
Shared secret between P_i and P_j	$g^{r_i r_j}$	$g^{r_i r_j}$	$g^{r_i r_j}$	$g^{r_i r_j}$
Secret DH value of P_i	r_i	r_i	r_i	r_i

The keying materials except the shared secrets $g^{r_i r_j}$ between a member P_i and the other members P_j (j=1,2…*n*. j≠i) as well as its secret DH value are replaced by new values at beginning of each session. From formula (6.2) and (6.3) we can see that the freshness of the temporary shared key K_{ij} between member P_i and P_j mainly depends on the nonce N_i. Since each session uses a different nonce, K_{ij} is refreshed. In case of group refreshment the new group key is protected by K_{ij}. Let us assume that group member P_i is successfully attacked during session 3. When the intruder breaks into the host of member P_i he/she can only access the keys and keying material of the current session, e.g. GK_3, K_{ij3}, N_{i3} , r_i and $g^{r_i r_j}$, but not any other keys and keying material of the former sessions except $g^{r_i r_j}$ and r_i. If the intruder wants to get GK_2, he/she has to recover K_{ij2}. To reconstruct K_{ij2}, N_{i2} is required. However, N_{i2} is protected by the former session key GK_1 (see message M_{J5} in Section 6.4.1 and message M_{L1} in Section 6.4.2), i.e. if the intruder wants to get GK_2, he/she must know GK_1. The latter, however, does not exist any more in the system at this time. Thus the intruder is unable to recover GK_2 with the keys and keying material of session 3. The same holds for GK_1. Therefore, we can conclude that even if an attacker intrudes a host during a session, he/she cannot decrypt previous sessions.

Resistance to known key attacks means that the disclosure of past session keys cannot be used to compromise the current session key. Here we again have to consider two cases. The first case considers a passive adversary who records the data exchanged over a communication link to analyze them later [70]. We assume that the passive adversary knows the past session keys and the nonce N used to derive the temporary shared key K_{ij} between P_i and P_j to protect the new group key (see messages M_{J5} and M_{L1}). To successfully reconstruct K_{ij} the passive attacker has also to know the shared secret $g^{r_i r_j}$ of P_i and P_j besides the nonce (see formula (6.2) and (6.3)). This is impossible, because the shared secret $g^{r_i r_j}$ is determined in P_i and P_j and never transmitted over a communication link. Thus the attacker is unable to derive K_{ij}. Without K_{ij} he/she cannot access the new group key, i.e. compromised past session keys cannot be used by a passive adversary to compromise future session keys.

The second case relates to an active adversary who tries to modify data on the communication link [70]. We consider the situation when the token holder P_i refreshes the group key by either sending M_{J5} or M_{L1} to the remaining group members. We further assume that the attacker knows the old group key GK_{old} by some means, so that he/she could capture M_{J5} or M_{L1} and modify the message with GK_{old} as follows:

P_i (Token holder)		Active adversary		Receiver P_1
HDR, $GK_{old}\{ID_i, N_i\}$, $K_{i1}\{VK, GK_{new}\}\ldots$	⟶	HDR, $GK_{old}\{ID_i`, N_i`\}$, $K_{i1}\{VK, GK_{new}\}\ldots$	⟶	HDR, $GK_{old}\{ID_i`, N_i`\}$, $K_{i1}\{VK, GK_{new}\}\ldots$

The attacker replaces ID_i and N_i by another identity $ID_i`$, $N_i`$. The group members, however, receiving the forged messages will generate a different temporary shared key $K_{ij}`$ which fails when authenticating message $\{VK,GK_{new}\}$. Thus the group members will be aware of the ongoing attack, i.e. an active adversary cannot impersonate one of the protocol parties using past session keys.

Finally we compare the security properties of VTKD with that of TGDH and Rodeh's protocol. The security features of the three protocols are depicted in Table 6.1.

Table 6.1: Security features of VTKD, TGDH and Rodeh's protocol

Security features	Rodeh	TGDH	VTKD
Key authentication	**No**	**No**	Yes
Forward confidentiality	Yes	Yes	Yes
Backward confidentiality	Yes	Yes	Yes
Collusion freedom	Yes	Yes	Yes
Perfect forward secrecy (long-term key)	Yes	Yes	Yes
Perfect forward secrecy (host hacked)	Yes	**No**	Yes
Resistance to known key attacks	**No**	Yes	Yes

TGDH and Rodeh's protocol do not provide *key authentication*, since they do not possess an entity authentication function which is the essence of key authentication [70].

TGDH incompletely supports *perfect forward secrecy*. In case of a hacked host TGDH is not capable to keep perfect forward secrecy, because the intermediate keys generated by the sponsor are transmitted over a so-called public and authentic channel. They can readily be intercepted by the adversary. If the adversary compromises one of the group hosts and gets access to the private DH value of that host, he/she can generate all group keys used in the previous sessions.

Rodeh's protocol is subject to *known key attacks*, since the tree leader encrypts the new group key with the old group key and multicasts it to the group members for a join event.

6.6 Performance analysis

This section evaluates the performance of VTKD in comparison with other decentralized group key exchange protocols. We consider the key distribution protocol of Rodeh et al. and TGDH which are considered the most efficient group key distribution and agreement protocol, respectively. For the comparison, we first measured benchmarks of cryptographic algorithms under the platform Intel Xeon 2.6 GHz processor running Linux by utilizing the OpenSSL [138] as the cryptographic library. The benchmarks are summarized in *Appendix A.* These benchmarks were then applied to evaluate the performance of the three protocols.

6.6.1 Member authentication

In VTKD each participant has to be authenticated by a group member with the standard protocol IKEv2 before joining the group. Since other both protocols do not possess an authentication function, we can only specify the authentication cost of VTKD here. The authentication cost comprises the standard IKEv2 computation cost (messages M_{J1}~M_{J4}) for mutual authentication and the n DH agreements performed by the invitee on receipt of message M_{J3} to get the shared DH secrets with other members. The standard IKEv2 computation cost consists of 2 RSA signatures, 2 validation signatures, 4 symmetric crypto operations and 4 hashes. It is obvious that the n DH agreements are the most computation intensive operations of the authentication phase. Basically, the new member does not have to compute these n DH-agreements in real-time. This can be done in the background, since the new member will apply these shared DH-secrets only when it becomes the token holder to establish the temporal secure channels. Therefore, the authentication delays are evaluated for two cases: (1) for computing the n DH-agreements in real-time and (2) for calculating the n DH-agreements in the background. As shown in Figure 6.9 the authentication delay for the former case increases quasi linearly with the group size, while it nearly keeps unchanged for the latter case.

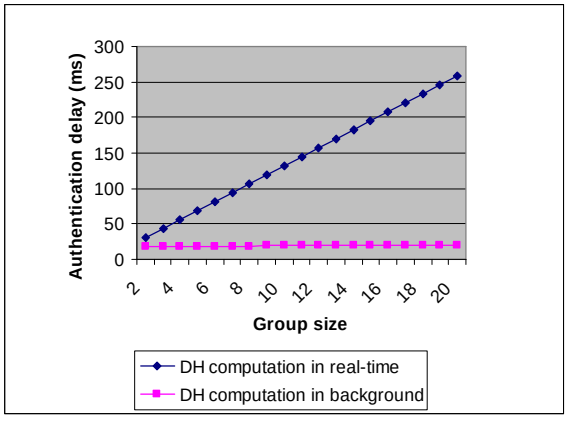


Figure 6.9: Authentication delay of VTKD

6.6.2 Group key renewal

Now we compare the key renewal cost with the other protocols. A widely accepted criterion to evaluate the efficiency of group key management protocols is the group refreshment delay. It refers to the latency required to renew the group key at all members when a refreshment event occurred (see Figure 6.10). The group key refreshment delay should be as small as possible, since in this period the real-time video/audio communication of the group is interfered due to the fact that some members might apply the new group key, while others are still using the old one. In contrast, no such strict demands are posed on the authentication delay, because during authentication all members are still using the old group key for real-time data exchange.

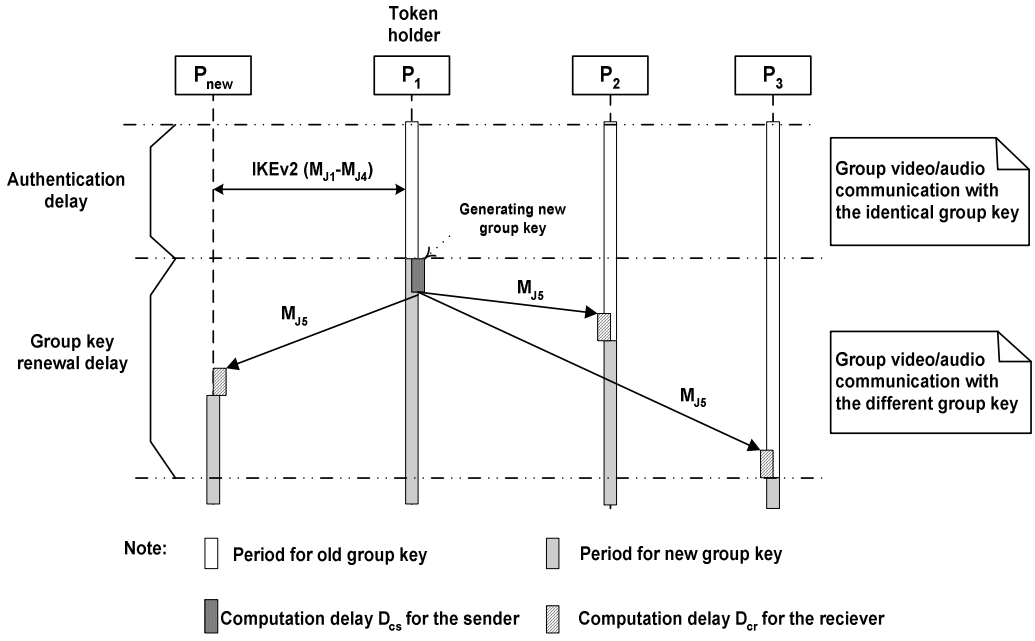


Figure 6.10: Impacts on the real-time group communication caused by authentication and group key renewal delay

The group key refreshment delay comprises the communication delay and the cryptographic computation delay. It is determined by the following formula:

$$D_{gkr} = \max(D_{cs} + D_{com} + D_{cr}) \tag{6.4}$$

where D_{gkr} is the group key refreshment delay, D_{cs} the cryptographic computation delay of the sender, D_{com} the group communication delay caused by the underlying group communication protocol, and D_{cr} the cryptographic computation delay of the receiver. Here the sender stands for the tree leader in Rodeh's protocol, the sponsor in TGDH, and the token holder in VTKD, respectively. The receiver corresponds to the participants including the

new member and the subtree leader of Rodeh's protocol, and the participants including the new member in TGDH and VTKD.

Cryptographic computation delay

The cryptographic computation delay directly depends on the computation cost of the protocol. Table 6.2 summarizes the computation cost of the considered protocols by indicating the number of cryptographic operations they carry out. The comparison shows that the protocols use asymmetric and symmetric cryptographic operations differently. TGDH at the one edge applies more intensively asymmetric cryptographic computations while VTKD at the other edge uses mainly symmetric operations. Since asymmetric cryptographic computations, as known, are much slower than the symmetric operations, the resulting total computation cost of VTKD is lower than that of the other two protocols. For example, the computation cost of one DH agreement corresponds to approximately 10000 HMAC calculations and symmetric encryptions of the group key (16 byte size) or 1250 HMAC calculations for creating the temporal secure channel (209 bytes) based on the benchmarks of cryptographic algorithms shown in *Appendix A*. Assuming a group size of 100 members, only about 102 HMAC calculations and symmetric encryptions (16 byte) as well as 202 HMAC calculations (209 byte) are required to renew the group key in VTKD.

Table 6.2: Computation cost for the group key renewal

Protocols	Operation	Members	Computation cost					
			Asymmetric operations			Symmetric operations		
			DH agreement	RSA signature[2)]	RSA verification[2)]	HAMC and encryption (16 byte)	HAMC and decryption (16 byte)	HAMC (209 byte)[4)]
Rodeh	Join	Tree leader	1	-	-	2	-	-
		New member	1	-	-	-	1	-
		Participants	-	-	-	-	1	-
	Leave	Tree leader	$\log_2 n$[1)]	-	-	$\log_2 n$	-	-
		Subtree leader	1	-	-	-	1	-
		Participants	-	-	-	-	1	-
TGDH	Join	Sponsor	$2 \log_2 n$	1	-	-	-	-
		New member	$2 \log_2 n$	-	1	-	-	-
		Participants	$1 \ldots 2\log_2 n$	-	1	-	-	-
	Leave	Sponsor	$2 \log_2 n$	1	-	-	-	-
		Participants	$1 \ldots 2\log_2 n$		1	-	-	-
VTKD	Join	Token holder	-	-	-	n	-	2n
		New member	-	-	-	-	1	-
		Participants	1	-	-	-	2	2
	Leave	Token holder	-	-	-	n	-	2n
		Participants	-	-	-	-	2	2

Note: 1) n is the number of group members.

2) RSA signature in TGDH is used to support message authentication rather than member authentication [123].

3) The computation costs of Rodeh and TGDH listed in the table are their best case when the key tree is balanced. For an imbalanced key tree Rodeh and TGDH need more computation for a rekeying event.

4) This HMAC computation is used to construct the temporal secure channels between the token holder and other members according to formula (6.2) and (6.3). Here we assume that the sizes of the shared DH value $g^{r_i r_j}$, Nonce N, entity identity ID are 128, 16, and 32 bytes respectively.

Applying the benchmarks of the cryptographic algorithms from *Appendix A* to Table 6.2 we can compute the cryptographic computation delay for the protocols, i.e. $max(D_{cs}+D_{cr})$. The results are listed in Table 6.3. It shows that VTKD causes less computation delay than the other two protocols for joining and leaving.

Table 6.3: Computation delay for group key renewal (ms)

	Rodeh	TGDH	VTKD
Join	25	$5.16+50*\log_2 n$ [1)]	$12.5+(n+2)*10^{-3}+(2n+2)*8*10^{-3}$
Leave	$12.5*(\log_2 n+1)+(\log_2 n+1)10^{-3}$	$5.16+50*\log_2 n$	$(n+2)*10^{-3}+(2n+2)*8*10^{-3}$

Note: 1) n is the number of group members.

Communication delay

The communication delay for a group key renewal depends on the number of communication rounds needed to complete the renewal procedure and on the duration of the communication rounds. For a fair comparison, we assume here that the three protocols run on top of the group communication protocol GCP. According to [8] the delay D_{c1} for one communication round in a LAN setting can be estimated as follows:

$$D_{c1} = 8.71*n-8.63+0.0015*b \quad (6.5)$$

where n is the number of group members and b the size of the rekeying message in bytes. VTKD and TGDH require only one communication round to accomplish the group key renewal for joining and leaving, whereas the Rodeh protocol needs two communication rounds each.

Group key renewal delay

Based on Table 6.3 and formula (6.5), we can now determine the group renewal delay using formula (6.4). The resulting delays for the join and the leave procedures are depicted in Figure 6.11 and 6.12.

In Figure 6.11 and 6.12 the dashed line represents the group communication delay of GCP. It shows for the join procedure that VTKD does not add a larger delay to that of the group communication, whereas the dashed line almost overlaps with the VTKD line for the leave procedure (see Figure 6.12). This means that the group key refreshment delay of VTKD caused by the leave event is nearly equivalent to the group communication delay. This is because all cryptographic computations used for leaving in VTKD are symmetric operations. The delay caused by these operations is negligible compared to the group communication delay.

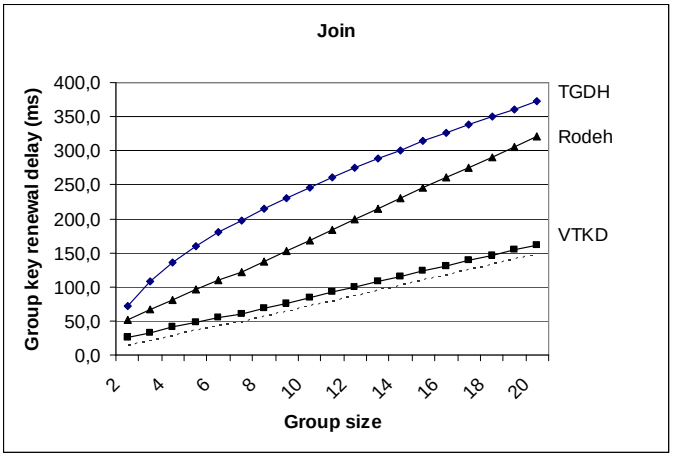


Figure 6.11: Key refreshment delay of the join procedure for the compared protocols

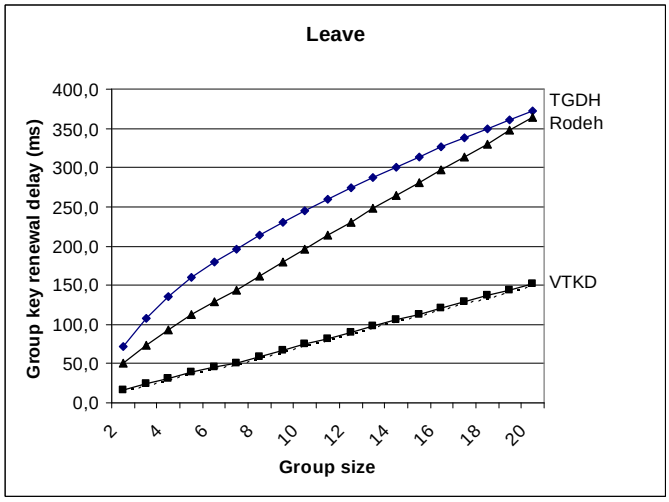


Figure 6.12: Key refreshment delay of the leave procedure for the compared protocols

The comparison of Figures 6.11 and 6.12 further shows that VTKD offers a better performance related to the group key refreshment delay for both procedures than the other both protocols. The reason is that VTKD needs only one communication round to complete the protocol and mostly uses symmetric cryptographic operations. TGDH is the most expensive one, although it can renew the group key in one communication round like VTKD. It applies a lot of asymmetric cryptographic operations to generate the new group key for each member. To renew the group key in TGDH, the sponsor has to generate a set of new intermediate keys with $2log_2n$ DH-agreements which are fairly intensive computation operations

compared to symmetric cryptographic operations. When receiving the intermediate keys each member individually computes the new group key with its own share causing the same computation burdens as for the sponsor. Rodeh's protocol is the second expensive one, although it mainly exploits symmetric cryptographic computations like VTKD. This is because it needs two communication rounds to accomplish the group key renewal, one more communication round than VTKD and TGDH need. To renew the group key in Rodeh's protocol, the tree leader generates the new group key and sends it to the subtree leaders in the first communication round over the underlying group communication platform. In the second communication round the subtree leaders deliver the new group key to their own members, i.e. the group key refreshment delay of Rodeh's protocol will be greater than the double value of the group communication delay introduced by the underlying group communication protocol.

As shown above, TKD presents the best performance of three protocols. However the rank between TGDH and Rodeh's protocol needs further discussion. In the above comparison the performance of Rodeh's protocol outweighs that of TGDH, because the cryptographic computation delay plays more role than the communication delay in determining the group key refreshment delay under the platform Intel Xeon 2.6 GHz. It is worth noting that a reverse conclusion may be drawn if the comparisons are based on the more power device, because the stronger power a device, the less computation delay introduced by the asymmetric algorithms.

6.6.3 Bandwidth overhead

The bandwidth overhead depends on the message size of the protocol, i.e. how many bandwidth a protocol consumes. Many centralized group key distribution protocols such as LKH [120] apply a key tree structure, so that the group key server can update the group key using a message size of $O(log_2 n)$ symmetric keys. This is of particular significance for very large groups (e.g. one million members). Thus the rekeying message can be delivered in one packet. Some decentralized group key distribution protocols like Rodeh and TGDH follow the same principle to achieve a small rekeying message size, $O((log_2 n)^2)$ symmetric keys for Rodeh and $O(log_2 n)$ asymmetric keys for TGDH. This is, however, achieved at the cost of two communication rounds in the Rodeh protocol and of $O(log_2 n)$ asymmetric cryptographic computations in TGDH. This is the reason why they are slower than our scheme. In VTKD the rekeying message size is $O(n)$ symmetric keys. For a group of 100 member peers, these are about 4 Kbyte. This can be transmitted without any problem in one UDP packet. Therefore bandwidth consumption for key renewal is not an issue for small group settings at all. In contrast, the group key renewal delay is the critical point for real-time applications.

6.6.4 Theoretical upper bounds of the group size

To know up to what group size the VTKD protocol is acceptable, it is necessary to estimate the theoretical upper bounds of the group size that VTKD can support. This estimation is made on two conditions: (1) the key renewal delay of VTKD should fall below that of the compared protocols and (2) VTKD accomplishes the group key renewal in one communication round.

The group key renewal delay of VTKD, TGDH, and the Rodeh protocol can be determined based on Table 6.3 and formula (6.5) as follows.

For a joining event:

$$D_{VTKD-J} = 12.5 + (17*n + 18)*10^{-3} + 8.71*n - 6.83 + 0.0015*36*n \quad (6.6)$$

$$D_{TGDH-J} = 5.16 + 50*\log_2 n + 8.71*n - 6.83 + 0.0015*128*\log_2 n \quad (6.7)$$

$$D_{Rodeh-J} = 25 + 2*(8.71*n - 6.83 + 0.0015*36*\log_2 n) \quad (6.8)$$

For a leaving event:

$$D_{VTKD-L} = (17*n + 18)*10^{-3} + 8.71*n - 6.83 + 0.0015*36*n \quad (6.9)$$

$$D_{TGDH-L} = 5.16 + 50*\log_2 n + 8.71*n - 6.83 + 0.0015*128*\log_2 n \quad (6.10)$$

$$D_{Rodeh-L} = 12.5*(\log_2 n + 1) + 2*(8.71*n - 6.83 + 0.0015*36*\log_2 n) \quad (6.11)$$

Thus condition (1) can be expressed through the following four formulas:

$$D_{VTKD-J} \le D_{TGDH-J} \quad (6.12)$$

$$D_{VTKD-J} \le D_{Rodeh-J} \quad (6.13)$$

$$D_{VTKD-L} \le D_{TGDH-L} \quad (6.14)$$

$$D_{VTKD-L} \le D_{Rodeh-L} \quad (6.15)$$

These formulas can be now used to determine the upper limit of the VTKD group size. The solutions for formula (6.12) and (6.14) are n=9205 and n=9402, respectively, whereas formula (6.13) and (6.15) remain true for any group size, i.e. VTKD is always more efficient than the Rodeh protocol if VTKD completes the key renewal in one communication round.

Condition (2) means that the size of the rekeying messages is always less than the maximum size of an UDP packet payload, which is 65508 bytes (i.e. the maximal length of an IP packet minus the head length of IP and UDP). Thus the upper bounds of group size can be determined by the following formula:

$$hdr + (n+1)*(sk+h) + ak \le 65508 \quad (6.16)$$

where n is the number of group members, while *hdr*, *sk*, *h*, and *ak* correspond to the size of HDR, the symmetric key, the hash value and the asymmetric key, respectively. Their corresponding typical values are 20 bytes, 16 bytes, 20 bytes and 128 bytes. Formula (6.16) holds as long as n is smaller than 1814.

To sum up, VTKD is more efficient related to the key renewal delay than other key exchange protocols as long as the group size does not exceed 1814 members.

6.7 Implementation

In the above section we have analyzed the performance of the VTKD protocol in theory. This section presents experimental results of its integration into the BRAVIS system. Beforehand we introduce the solution to a crucial issue associated with this integration, i.e. how to maintain the virtual synchrony feature of the GCP protocol during the rekeying phase.

6.7.1 Integration

The BRAVIS system has been constructed in a module-based manner. Each module realizes a specific function so that the entire system is easy to maintain and expand. New functions, e.g. group key management, can be implemented as new modules in the system without the need to greatly change the existing modules. Each module usually uses one process. The communication between these processes (i.e. modules) is realized by means of *message queues* and a *shared memory* which are standard functions under operation systems Linux and UNIX. *Message queues* ensure the reliable and ordered message delivery among modules. Different processes can write data into and read data from the *shared memory*. Block mechanisms, such as *semaphores*, are used to safeguard the consistence of data in the *shared memory* so that all modules can keep the identical pace to run after accessing the data in the shared memory.

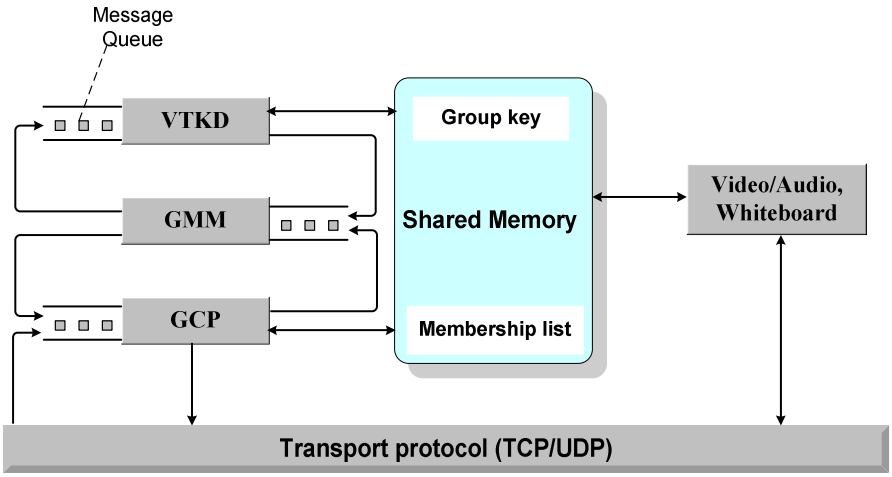


Figure 6.13: Integration of VTKD protocol

The VTKD protocol is integrated into the BRAVIS system in the same way as these existing modules. It is implemented in a single module, named VTKD which provides no more function than the designed group key management. As shown in Figure 6.13, the VTKD module communicates with other modules via *Message queues* and the *shared memory* as well. The GCP module, which implements the GCP protocol (GCP), is aware of changes of group composition (join, leave, and failure of peers). When such change occurs, the GCP module updates the membership list in the *shared memory*, translates these events into corresponding *JOIN* or *LEAVE* primitive, and notifies them to the upper modules through the *group management module* (GMM) using *Message queues*. After reading *JOIN* or *LEAVE* primitive from the *Message queue*, the VTKD module in the token holder generates the new group key and securely delivers it to the group. Accordingly every VTKD module in the group equally refreshes the group key in the *shared memory* so that members can make confidential talks using the same key. The detailed information about the interaction between VTKD and GCP, and the state machine of the VTKD protocol are introduced in *Appendix B*. The other modules, such as video/audio manager, Whiteboard, which need the current group key to encrypt their data, can access the *shared memory* for this.

Note that the authentication messages and rekeying messages in the VTKD protocol separately take advantage of the different transport platforms for their transimission. Four authentication messages (message M_{J1}~ M_{J4}) are transmitted over the well-know UDP port rather than over the group communication protocol GCP used in BRAVIS. Such implementation is in compliance with the IKEv2 standard. The reliable delivery of these four messages is achieved by these widely used measures such as several times resending after a timeout which have been specified in IKEv2. Two rekeying messages (message M_{J5} and M_{L1}) rely on the GCP protocol for their reliable and ordered delivery.

6.7.2 Maintaining the consistency of signaling data

In the above subsection we have figured out the blueprint how to integrate the VTKD protocol into the BRAVIS system. Essentially one vital issue related to this integration has not been touched. As shown in Section 6.6.2, media data (video/audio) are perturbed during the rekeying phase because it is impossible to install the new group key at the same time point for asynchronous hosts. The signaling data face with the same problem. However, this issue imposes different impacts on the media and signaling data. In general, the media data are tolerant for such interference. Some video/audio frames are lost due to incorrect decryption using a different key with the sender during the rekeying period. But this does not lead to the termination of a conference. It just degrades video and audio quality. More efficient a group management protocol, shorter this interference period. On the other hand, such interference on the signaling data is not allowed. Members can not correctly decrypt the signal-

ing data if this interference happens. This completely destroys the virtual synchrony feature of the underlying protocol GCP. All members are therefore unable to equally update the signaling data used for the control of the conference. As result the conference may be forced to terminate by each rekeying event. This is obviously unacceptable for users. Therefore an appropriate solution to keep the consistency of the signaling data during the rekeying phase is required for the successful integration of the VTKD protocol.

The intuitive approach is to synchronize the clocks of all hosts. So the hosts can agree upon a unique time point to enforcing the new group key. Thereby the above mentioned problem is trivially solved. Although several protocols, such as the Network Time Protocol (NTP) [140], and the Digital Time Synchronization Protocol (DTSS) [141] are available for this purpose, their deployments are confined by their expensive cost and burdensome configuration in each host. Their basic idea is that some number of computers which synchronize with the standard timescale Coordinated Universal Time (UTC) acts as primary time servers in a network. They are further used to synchronize a larger number of secondary servers and clients residing in a common network. Thus a considerably large number of time servers have to be deployed for the clock synchronization purpose. At present such approaches are only applied to some closed communities, not to the whole Internet. It is unrealistic that all hosts in the Internet could be synchronized in the near future, since the number of hosts in the Internet is so huge that a limited number of time servers available today are unable to serve all hosts in the Internet for clock synchronization.

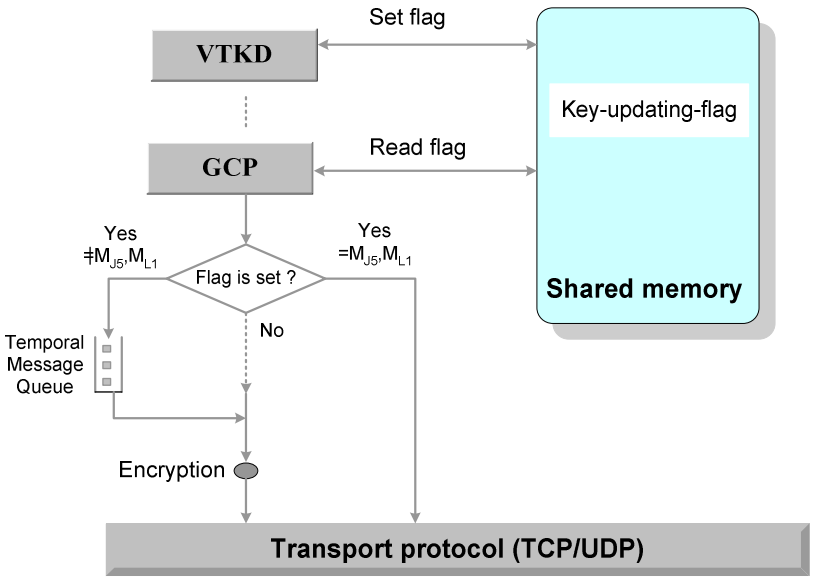


Figure 6.14: Maintaining the consistency of the signaling data

A simple and effective approach was implemented by Krauz [142] to address this issue. It does not need to synchronize the clocks of all hosts. This approach consists of two measures to preventing the signaling data from the inconsistency among the group: a *key-updating-flag* in shared memory and a *temporal message queue* (see Figure 6.14). The former notifies the GCP module the beginning and ending of the key updating phase. The later is used for the GCP module to temporally save the signaling data from the upper modules (QoS, Floor control module etc.) during the key updating phase.

This approach works as follows. When the VTKD module receives the primitives (*JOIN* or *LEAVE*) indicating the group composition change, it activates the *key-updating-flag* in the shared memory to mark the beginning of the key updating phase. Every time the GCP module processes signaling data, it checks whether the *key-updating-flag* is set. For the active flag, the GCP module further examines whether the received signaling data are messages M_{J5} or M_{L1} (from the VTKD module). If so, they are directly sent to network since the content of these two messages (new group key) have already been separately encrypted using the temporal secure channels (see Section 6.4). Thus there is no need to encrypt them again. If the signaling data are other than messages M_{J5} and $M_{L1,}$ the GCP module pushes them into the *temporal message queue*. This means that the signaling data except VTKD messages are blocked in the *temporal message queue* from the beginning of the key updating phase. The VTKD module deactivates the *key-updating-flag* in the shared memory to mark the ending of the key updating phase in response to the receipt of messages M_{J5} or M_{L1} which are used in the VTKD protocol to refresh the group key. Once the GCP module detects the *key-updating-flag* inactive, the signaling data saved in the *temporal message queue* are first processed by the encryption function, which now has installed the new group key. The newly arriving signaling data will be processed until the t*emporal message queue* is empty. The essence of this approach is to postpone the procession of signaling data until the new group key in all members is installed. All members can correctly decrypt the received signaling data under the new group key. As a result, the inconsistency of signaling data is avoided and the important virtual synchrony feature of the GCP protocol is retained. In theory, this approach could be used in the audio/video module to remove the interference during the rekeying phase. However, it first does not well meet the strict real-time requirement of video/audio communication since audio/video frames have to be blocked in the temporal message queue until the ending of the rekeying phase. Secondly a message queue with considerably large size is needed to temporarily deposit the audio/video data. This is an issue especially for high bit rate video data. Therefore this approach used for achieving the consistency of the signaling date should not be recommended to be applied to the video/audio data.

6.7.3 Experimental results

The performance of the VTKD protocol running on the BRAVIS system has been measured in the labor of the group computer and communication network of BTU. The computers in the labor are connected via LAN. They have the identical hardware and software configurations as follow:

- *Processor:* Dual Intel Xeon 2.66 GHz
- *Memory:* 2 GByte RAM, 512 KByte Cache, 1 GByte Swap
- *Operation system:* SuSE Linux 9.2, Kernal Version 2.4.xx
- *Crypto library:* OpenSSL Version 0.9.7e

The group key renewal delay of the VTKD protocol was measured with group sizes of 2, 3, and 8 members, respectively. For each case, the measurement was repeated ten times. The test results varied from one measurement to another one. This is mainly caused by the GCP protocol. The GCP protocol used to deliver the rekeying messages M_{J5} and M_{L1} is also based on token mechanism. Only the token holder of the GCP protocol can send the data to the group for ordered delivery purpose. When the token holder of the GCP protocol and the token holder of the VTKD protocol are overlapped, the rekeying messages M_{J5} and M_{L1} can be immediately sent to the group by the GCP protocol. This is the best case for the group key renewal. The worst case for the group key renewal occurs when a member becomes the token holder of the VTKD protocol just after it hands over the token of the GCP protocol to the neighbor. In this case, the rekeying messages M_{J5} and M_{L1} are not delivered to the group until the member holds the token of the GCP protocol again. Therefore we use the average of ten measured values as the final results to represent the group key renewal delay. They are summarized in the following table [142]:

Table 6.4: Experimental results (ms)

Group size	Join		Leave	
	Experimental results	Theoretical evaluation	Experimental results	Theoretical evaluation
2	25.65	23.25	-	-
3	31.25	32.03	20.89	19.53
8	74.25	75.94	61.03	63.44

From Table 6.4, we can observe that the experimental results are quite close to the corresponding theoretical evaluation values. This confirms the correctness of the theoretical evaluation for the group key renewal delay of the VTKD protocol.

6.8 Summary

The key refreshment represents the basis for confidential talks in small dynamic closed peer groups. Many collaborative applications, in particular business applications, require this to confidentially communicate via the Internet. Scalability is not the key issue for these group settings [115]. Efficiency and security are the primary concerns for their use.

This chapter presented a simple distributed group key distribution protocol VTKD which uses an intuitive and straightforward principle, i.e. token mechanism, for the key refreshment. It mostly uses symmetric cryptographic operations and needs only one communication round. This leads to a significantly lower key renewal delay compared to existing key distribution and key agreement protocols. It is especially appropriate for applications which except key management and encryption/decryption simultaneously run other time and resource consuming procedures such as media data decompression like in a peer-to-peer multiparty video conference. In addition, VTKD is the unique approach among the compared protocols to fulfill all security demands required by a secure group communication.

Key tree based protocols like LKH have been proven to be an appropriate solution for a centralized group key management, also for small groups [120]. Several distributed protocols like TGDH and the Rodeh protocol borrowed the key tree concept. However, the efficiency of centralized key tree approaches does not hold for the distributed key tree approaches when the group size is small. VTKD has proven more efficient for the targeted application of small dynamic peer groups than existing key tree based protocols. The further advantage of VTKD is that it provides a stable performance and needs less effort for its maintenance. In contrast, key tree based schemes like TGDH and the Rodeh protocol possess a fluctuating performance after many rekeying operations due to the unbalance of the key tree. To maintain a balanced key tree rebalance algorithms have to be applied which makes the protocols more complex and less practical.

The basis for the VTKD protocol is the use of a group communication protocol which assures virtual synchrony. Such protocols exist meanwhile and as shown in [143] they can be used for peer-to-peer audio/video conferences in areas like Middle Europe.

Chapter 7

A Novel Algorithm for Video Encryption

Employing the secure and efficient group key management protocol VTKD introduced in the previous chapter, the group members can agree upon a common group key used for encrypting their communication, including signaling data and media data (video, audio and whiteboard). While for signaling, audio, and whiteboard (text) encryption applicable encryption algorithms are available, there is still a lack of appropriate video encryption algorithms. Specific video encryption algorithms are strongly required in real-time multimedia communication to fulfill the strict timing requirements. In this chapter we present the video encryption algorithm *Puzzle* to encrypting video streams in software. It is fast enough to fulfill real-time constraints and provides sufficient security meanwhile. *Puzzle* is a video compression independent algorithm which can be easily incorporated into existing multimedia systems

7.1 Introduction

Significant advances in video compression technologies pave the way to economical storing of a digital video on storage constrained devices or efficient transmitting of a digital video over bandwidth-limited networks. Raw (uncompressed) digital video needs an extremely high transmission bandwidth; for example, digitized uncompressed PAL (Phase alternation line) resolution video (720x576 pixels at 25 frames per second) has a bit rate of approximately 248 Mbits/s [108]. Video compression techniques allow to representing these raw digital videos in significantly less bit rate without obvious loss of visible quality. This facilitates their storing and the transmission over networks. However, video compression standards like MPEG-1 [144], MPEG-2 [145] or H.26x [146], [147], are computationally intensive algorithms. Especially the video compression requires much more compute power than the decompression [148]. To meet the time constraints of interactive real-time applications, many multimedia systems implement video compression in hardware, e.g. on dedicated chips, while video decompression is implemented in software for cost reasons.

More recently developments in video compression and networking technologies make networked multimedia applications becoming increasingly popular, e.g. video on demand or video conferences. Due to the openness of public networks, confidentiality is one of the primary concerns for their commercial use. So video on demand (VoD) service providers, for example, wish that only their subscribers can watch the video programs to ensure their commercial profits. In a business video conference only the participating members are allowed to receive the video (and the audio) to protect the confidentiality of the negotiation. This issue is usually addressed by encryption. Only authorized parties possessing the decryption keys are able to access to the multi-media contents. The straightforward way is to encrypt the entire compressed video stream with a conventional cryptographic algorithm such as AES [59]. This is called *naive algorithm approach* [149]. This approach is simple to implement and easy to integrate into existing multimedia systems, since it is independent of certain video compression algorithms.

Conventional cryptographic algorithms mainly aim at encrypting text data. They are CPU-intensive and do not well meet real-time constraints when they are directly applied to encrypting large amounts of compressed video data, especially if a software implementation is used. Although the size of the compressed data is significantly smaller compared to that of the raw video data, it is still much larger than usual text data. For example, the bit rate of a MPEG-2 video stream typically ranges between 4 and 9 Mbps [108]. Nowadays, advanced computers are fast enough to encrypt a single channel MPEG-2 video stream in real-time using the naive algorithm approach. However, this evolution of the computer power does not completely eliminate the need to develop faster encryption algorithms for video data. Many multimedia applications such as video on demand and multiparty P2P video conferences always require specific algorithms for the video encryption because they usually support multi-channel video communication. The simultaneous encryption or decryption, respectively, of all streams causes a huge processing burden at the end systems. Specific encryption algorithms allow to alleviate these burdens and to enroll more users in the service. Therefore specific algorithms for the encryption of video data are highly desired. They should be efficient enough to process video data in real-time and secure enough to defend against possible attacks.

Since mid 90’s many research efforts have been devoted to designing specific video encryption algorithms. Several algorithms were proposed. These algorithms, however, are characterized by a considerable unbalance between security and efficiency. Some of them are efficient to fulfill the real-time requirements but with a limited security level, whilst others are vice versa strong enough to meet the security demands but with a limited encryption efficiency. Moreover, most of these algorithms are related to a certain video compression algorithm and implemented together in software. This makes them less practicable,

because today video compression algorithms are standardized and mostly implemented in hardware.

7.2 Related work

7.2.1 Video compression technologies

Video sequences inherently contain a high degree of spatial redundancy and temporal redundancy [150]. The former refers to the similarity among most neighboring pixels within a frame, whilst the latter refers to the similarity between adjacent frames. The objective of video compression is to remove both kinds of redundancy via a series of mathematic transformations using less data (true information) to represent the original video without a perceptual degradation.

The simplest video compression method is M-JPEG which strictly uses an intraframe coding. It compresses each frame independent of the others exploiting the still picture compression standard JPEG (Joint Photographic Experts Group) [151]. Thus only spatial redundancy is removed for an M-JPEG video. M-JPEG coding is performed by dividing a raw video frame into 8x8 blocks. These blocks are independently processed through three sequential procedures: Discrete Cosine Transformation (DCT), quantization, and entropy coding [148]. Each 8x8 block is first decomposed into spatial frequencies represented by 64 DCT coefficients using a DCT transformation so as to concentrate their energy into few coefficients which are suitable for the next steps to compress. Then the quantization process is applied to these DCT coefficients to reduce the number of bits by decreasing their precisions. One positive impact of this process on the last process (entropy coding) is that the number of zero DCT coefficients is drastically increased. The quantized DCT coefficients are scanned in a zig-zag order to maximize the number of consecutive zero coefficients. This is beneficial for the entropy coding to gain higher compression ratio. In the last step, these reordered coefficients are handled by the entropy coding, which comprises two lossless coding methods: *RLC* (Run Length Coding) and *VLC* (Variable Length Coding). RLC is a very simple data compression scheme in which a sequence of the same consecutive data values can be represented by a single data value and its repetitions. VLC takes advantage of the statistical properties of the encoded information to reduce the average number of bits. It assigns fewer bits to represent frequently occurring symbols and more bits to code rarely occurring symbols. Huffman coding [152] and arithmetic coding [153] are two typical representations in the context of VLC. On the decode side, the inverse operations are performed to reconstruct the video frames.

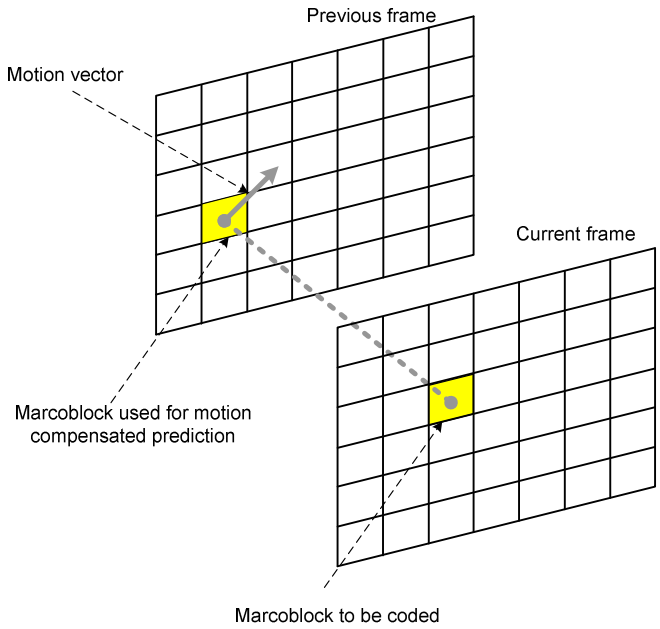


Figure 7.1: Motion compensated prediction

To gain better compression ratio, the broadly applied standards MPEG-1/2 and H.261/H.263 not only employ the intraframe compression algorithms similar to JPEG coding to eliminate spatial redundancy but also utilize interframe compression algorithms to reduce temporal redundancy. Widely used interframe compression algorithm is the motion compensated prediction technique [158] (see Figure 7.1) which assumes that a marcoblock (16x16 pixels) in the current frame may be found in the previous frame in view of the similarity of successive frames so that encoding of this marcoblock can be avoided. But such identical marcoblock may locate at different places in two frames. Thus a motion vector is defined to indicate the horizontal and vertical displacement between the block being coded and the block in the reference frame. The process of finding optimal or near-optimal motion vectors in an encoder is known as motion estimation. It is a computationally intensive operation. So the computational complexity of an encoder is usually much higher than that of a decoder because this operation is not involved in a decoder at all. It is apparent that some marcoblocks in current frame may have no closely matched marcoblock in the previous frame. These marcoblocks have to be encoded the same manner as in intra-frame coding. They are also called I-marcoblocks. There are three kinds of frames related to the motion compensated prediction: I (intra) frames, which are compressed independently without reference to the other frames like JPEG picture compression, P (predicted) frames, which are predicted from a previous I or P frame, and B (Bi-directionally predicted) frames, which are predicted from both previous and future I and/or P frames. H.261 utilizes the concept of I and P frames. The other standards take all three kinds frames into account.

7.2.2 Classification of specific video encryption algorithms

Existing video encryption methods have been comprehensively surveyed in [155], [156], where they are called *selective encryption* algorithms. This underlines the essence of these methods. They only partially encrypt the relevant video information to reduce the computational complexity by exploiting compression and perceptual characteristics. The relationship between selective encryption algorithms and video compression algorithms is a key factor to decide whether an encryption algorithm can easily be integrated into a multimedia system. Therefore we further classify the selective encryption algorithms into two categories according to their association with video compression algorithms: joint compression and encryption algorithms and compression-independent encryption algorithms.

Joint compression and encryption algorithms

The main idea of the *joint compression and encryption algorithms* is that encryption is applied to a certain step of the compression algorithm so that the output is significantly different from a video stream using a standard compression algorithm. The receivers cannot reestablish the original video without knowing the encryption key. Figure 7.2 illustrates the paradigm.

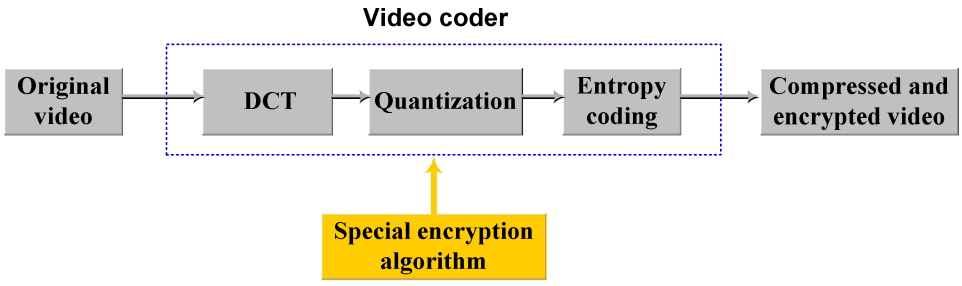


Figure 7.2: Principle of joint compression and encryption algorithms

A special encryption algorithm could be applied to one of the three encoding phases as depicted in Figure 7.2. This results in various joint compression and encryption algorithms. Zeng and Lei proposed a frequency domain scrambling approach [157] which is placed immediately after the DCT. It divides the DCT coefficients into blocks/segments and performs some or all of the following three operations: selective bit scrambling, block shuffling, and block rotation. This scheme is efficient and provides a considerable security level. It may though increase the bit rate of the compressed sequence up to 20 per cent [157]. There are two typical examples of special encryption algorithms after the quantization stage. First one is the ZigZag permutation algorithm proposed by Tang [158], which scans the 64 DCT coefficients by using a random permutation list instead of using the standard zig-zag order. This scheme adds a minimum computational overhead to the video encoding. However, it is vulnerable to known-plaintext and cipher text-only attacks [159]. Furthermore, an

increase of the size of the compressed video streams up to 46 per cent was observed in [159] compared with standard video stream. Another special algorithm is the real-time video encryption algorithm (RVEA) [160] introduced by Shi, Wang & Bhargava. It encrypts the selected most 64 sign bits of all DCT coefficients in a macroblock using conventional encryption algorithms. Although this approach can save 90 per cent of the encryption time compared to *naive algorithms*, it was shown that useful information content could be recovered from its output stream [161]. In the entropy coding phase, Wu and Kuo proposed the multiple Huffman tables scheme (MHT scheme) [161], [162]. It turns entropy coders into encryption ciphers using multiple statistical models to replace the only one statistical model in the standard video encoder. This approach utilizes 2^{14} pre-stored Huffman tables to encode the quantized DCT coefficients instead of one predefined fixed Huffman table used in the standard codec. This scheme encrypts video without sacrificing the compression performance for the Huffman coding. However, the bit rate will be increased by about 5 per cent for the arithmetic coding [162]. Moreover, as shown in [163], it is not strong enough to resist against the known/chosen plaintext attacks, since the number of the pre-stored Huffman tables is not sufficiently large.

The most important advantage of joint compression and encryption algorithms is that they add very little computational overhead to the original video codec, since they simply modify the original coding process by shuffling or selectively encrypting DCT coefficients, or using modified Huffman tables. The computational complexity of these operations is generally negligible as compared to that of a *naive algorithm*, so that the joint compression and encryption algorithms usually have a high encryption speed. However, as shown above, most of these algorithms are vulnerable to different kinds of attacks. For short, these schemes do not have a good trade-off between the security and efficiency.

It can be observed from the above introduction that the joint compression and encryption algorithms share a common shortcoming. They more or less decrease the compression efficiency of a video coder. Each encoding phase of a well-designed video coder has been optimized for high compression efficiency. To different extents, these special video encryption algorithms destroy the optimized structure of a standard video coder when they are embedded into one of the encoding phases. Another implied weakness is that they cannot be integrated into multimedia systems whose video codecs are implemented in hardware. Certainly, these approaches can be combined with multimedia systems implemented in software, but they completely destroy the modular design of the original codec. Appropriate modifications of the standard video codecs must be made to accommodate these schemes. Therefore, *joint compression and encryption* algorithms preclude the use of standard video codecs in multimedia systems.

Compression-independent encryption algorithms

The basic idea of *compression-independent encryption algorithms* (see Figure 7.3) is that compression and encryption are carried out separately. Only parts of the compressed video streams are encrypted with conventional algorithms taking the particular characteristics of the compressed video streams into account.

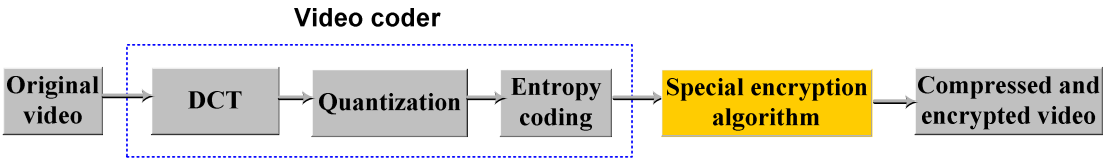


Figure 7.3: Principle of compression- independent encryption algorithms

So Spanos and Maples [164] and Li [165] exploit the fact that B- and P-frames are predicted from I-frames in interframe compression algorithms. Both proposed encryption approaches in which only the I-frames are encrypted. In theory it should prevent an eavesdropper without encryption key from the reconstruction of the original video. However, Agi and Gong [149] demonstrated that some scene contents are still discernible by directly playing back the selectively encrypted video stream on a standard decoder, since the unencrypted I-macro blocks in the B- and P-frames can be fully decoded without any information from the I-frames. Moreover, this approach did not achieve a significant computational reduction with respect to the total encryption, because the I-frames make about 30~ 60 per cent of an MPEG video [149]. Qiao and Nahrstedt [166] introduced the *video encryption algorithm* VEA in which half of the bit stream is encrypted with a standard encryption algorithm. This half stream is exclusive-ORed with the other half stream. The statistical analysis shows that MPEG video streams are almost uniformly distributed. VEA takes advantage of this special statistical behaviour of MPEG video streams to achieve the sufficient security level. However, the algorithm reduces the encryption load only by 47 per cent, since a half bit stream has to be encrypted with conventional algorithms.

The outstanding merit of the *compression-independent encryption algorithms* is that they completely eliminate the shortages of the *joint compression and encryption* algorithms with respect to the degradation of the video compression efficiency as well as the exclusion of a standard video codec. This is simply because the compression-independent encryption algorithms perform video encryption independent of the video compression procedures.

Criteria to designing a new specific video encryption algorithm

Inferred from the above analyses, a practicable and broadly applicable video encryption algorithm should fulfill the following requirements:

- *Easy integration.* The proposed scheme should be readily incorporated into an existing multimedia system independent of its implementation forms (in hardware or in software).
- *No impairment on compression efficiency*. The proposed scheme should not introduce any bit rate overhead to the compressed video sequence.
- *Good trade-off between security and efficiency*. The proposed scheme should have not only a high encryption speed but also an acceptable level of security.

As discussed above, the inherent features of joint compression and encryption algorithms cause the difficulty to satisfy with the first two conditions. Moreover, most joint compression and encryption algorithms do not possess a good trade-off between security and efficiency. Compression-independent encryption algorithms easily meet the first two requirements because the encryption and compression procedure are completely separate. So we prefer compression-independent encryption algorithms rather than joint compression and encryption algorithms when designing a new specific video encryption algorithm. Although several such kinds of algorithms are available, they do not achieve a noticeable encryption speed improvement compared to naive algorithms (only about a double speed-up). Moreover, some of them are not resistant against the simple perceptual attack, in which some contents in an encrypted video stream are discernible when it is played using a standard video player.

7.3 Principle of the *Puzzle* Algorithm

In this section we introduce an efficient and sufficiently secure video encryption algorithm, called *Puzzle*. The outstanding benefit of this scheme is the drastic reduction of encryption overhead for the high resolution video. We first give an overview on the basic idea of the *Puzzle* algorithm. After that the encryption steps are introduced in more detail. Finally the encoding and decoding procedures are presented.

7.3.1 Principle

The *Puzzle* algorithm is inspired by the children game puzzle which splits an entire picture into many small pieces and places them in disorder so that children cannot recognize the entire picture. When children play the game, they have to spend much time to put these pieces together to re-establish the original picture (see Figure 7.4).

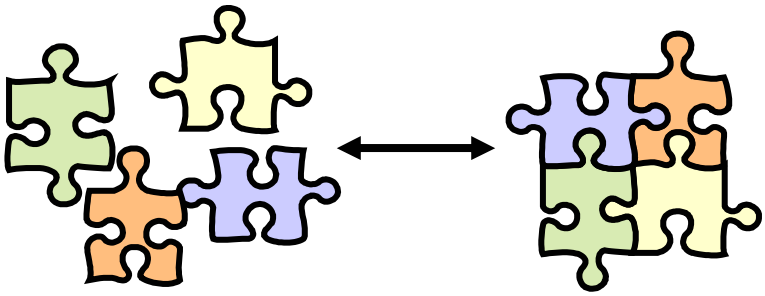


Figure 7.4: Puzzle game

Children usually reconstruct the original picture using or comparing known fragments of the picture and referring them to the accompanied original picture. We cannot therefore straightforwardly apply the game to encrypt a video frame. If we, however, modify the rules of the game in the following way, it will be nearly impossible for children to recover the original picture. The children should be only allowed to view the reverse side of the pieces so that they have to re-establish the picture without any hints to the original picture. It is manifested that $n!$ trials are required to re-establish the original, where n is the number of pieces. Basically, n needs not necessarily to be large. Assume that a picture is divided into 64 pieces, so the number of possible permutations is $64! = 1.27 \times 10^{89}$. It is unlikely that children reconstruct the original picture when having so many permutations. With this rule in mind we designed our *Puzzle* algorithm.

7.3.2 Encryption steps

Puzzle consists of two steps: (1) *Puzzling* the compressed video data of each frame and (2) *Obscuring* the puzzled video data. In step (1) the video data are partitioned into many blocks which are randomly shuffled afterwards. Step (2) corresponds to the turning over the blocks to the reverse side.

Step 1: Puzzling

A compressed video frame is puzzled by *partitioning* the frame into n blocks of same length b and *disordering* these blocks according to a random permutation list.

Partitioning

Given a L bytes long frame (excluding the frame header) of compressed video data $\boldsymbol{V}$ ($v_1 v_2 ... v_L$). The partitioning of the compressed video data $\boldsymbol{V}$ of length L into n blocks of the same length b is a typical factoring problem, i.e. $L = n \times b$. This problem is easy to solve if one of two variables (n,b) is assumed as constant. Unfortunately, we cannot solve this problem this way. If we fix the value of b, the value of n may become very large in some

frames or very small in other ones, since the length L varies for each frame. On the other hand, a too large value of n causes a larger computation overhead when exchanging the blocks. If the value of n is too small the scheme can be easily broken. To solve the problem we put some constraints on the variables n,b. The length of a block b should be $b=2^m$, where m is an integer. The value of n is only allowed to vary in the range from mb to $2mb$, whereby mb is a predefined constant number. It indicates that the compressed video data $\boldsymbol{V}$ should be at least split into mb blocks.

Using these constraints, the value of m can be uniquely determined by the following formula:

$$mb \le L / 2^m < 2mb \ . \tag{7.1}$$

The length of a block is given through $b=2^m$. The actual block number n can be calculated by the following formula:

$$n = \begin{cases} pn & \text{if } pn \text{ is even} \\ pn-1 & \text{if } pn \text{ is odd} \end{cases} \tag{7.2}$$

Where pn is the quotient of L/b. Formula (2) makes the value of n always an even number. This operation is necessary to disorder the blocks in the next step. With formula (7.1) and (7.2), the product of n and b might be unequal to the video frame length L when pn is odd or the remainder of L/b is unequal to zero. The difference between both is determined using the following formula:

$$d = L - n \times b \tag{7.3}$$

Formula (7.3) implies that the d bytes video data at the beginning of the video frame will be excluded from the disordering procedure.

Disordering

The basic idea for the disordering of the blocks is that the n blocks of compressed video data $\boldsymbol{V}'(v_{d+1}v_{d+2}\ldots v_L)$ are divided into two equal parts: an upper and the lower one. Each consists of $n/2$ blocks. Both parts are interchanged in accordance with a permutation list $\boldsymbol{P}=p_1p_2\ldots p_{n/2}$. This permutation list should be derived from a random sequence to resist an attacker to guess the original position of the blocks. We exploit a stream cipher with an key $\boldsymbol{K}$, such as SEAL [167] or AES-CTR [168], to generate $\ell+d$ bytes of a random sequence, called key stream $\boldsymbol{S}$ ($s_1s_2\ldots s_\ell\, s_{\ell+1}\, s_{\ell+2\ldots}\, s_{\ell+d}$), for each video frame. Since the values of the key stream $\boldsymbol{S}$ vary for each video frame, the permutation lists of different frames are distinct. Note that only first ℓ bytes of key stream $\boldsymbol{S}$ are used to produce the permutation list, the remaining d bytes of key stream $\boldsymbol{S}$ will be exploited in the obscuring step. The algorithm to generate the permutation list is depicted in Figure 7.5.

algorithm: Permutation list generation
input: Key stream $\boldsymbol{S}=s_1 s_2 \ldots s_\ell$,
n --number of blocks in the compressed video data $\boldsymbol{V}'$
output: Permutation list $\boldsymbol{P}=p_1 p_2 .. p_{n/2}$.

begin
Let $\boldsymbol{A}$ be an auxiliary sequence $\boldsymbol{A}=a_1 a_2 \ldots a_{n/2}$, its value of an element is
$a_i = i + n/2,\ 1 \le i \le n/2;$
Define $\boldsymbol{D}$ as another auxiliary sequence which is used to temporarily save the value selected from the key stream $\boldsymbol{S}$;
for $i=1$ to ℓ **do** */* Make the value of every element in* $\boldsymbol{S}$ *ranging from 1+n/2 to n. */*
if $((s_i \bmod n) \le n/2)$ $s_i = (s_i \bmod n)+n/2$;
else $s_i = s_i \bmod n$;
end if ;
Put s_i in the auxiliary sequence $\boldsymbol{D}$ without repetition;
Extract s_i from the sequence $\boldsymbol{A}$ and build sequence $\{\boldsymbol{A}\text{-}\boldsymbol{D}\}$;
end for;
$\boldsymbol{P}=\boldsymbol{D}\|\{\boldsymbol{A}\text{-}\boldsymbol{D}\}$; */* Get the permutation list* $\boldsymbol{P}$, || *denotes the append operation. */*
end

Figure 7.5: Permutation list generation algorithm

Using the permutation list we can generate the temporary cipher text $\boldsymbol{T}=t_1 t_2 \ldots t_{L-d}$ from the video data $\boldsymbol{V}'=v_{d+1} v_{d+2} \ldots v_L$ by swapping the i^{th} block of the upper part with the p_i^{th} block of the lower part of $\boldsymbol{V}'$. Figure 7.6 gives an example of this disordering process. It is assumed that a frame $\boldsymbol{V}'$ is split into 256 blocks $B_1 B_2 \ldots B_{256}$. The permutation list derived from the key stream $\boldsymbol{S}$ is $\boldsymbol{P}$= {256, 213, 216 … 130}.

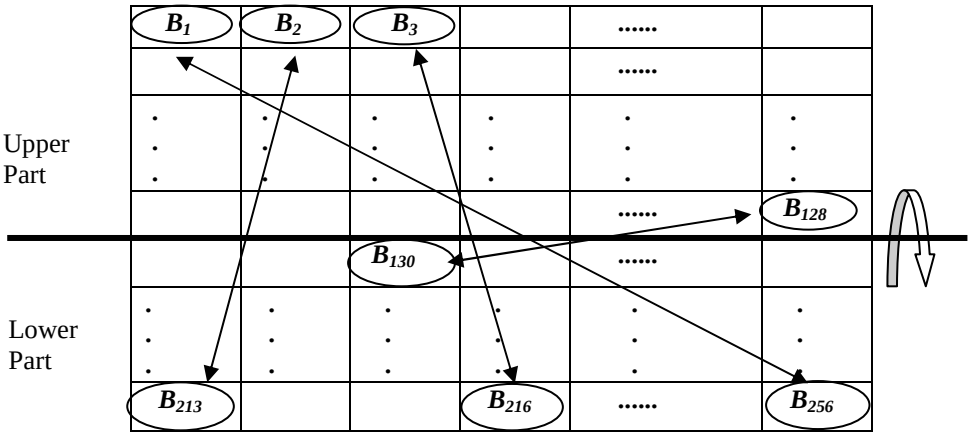


Figure 7.6: A Puzzle scenario

Step 2: Obscuring

The temporary cipher text ***T*** is obscured using a light-weight encryption. The basic idea is to encrypt only a small portion of ***T*** (first ℓ bytes) with a stream cipher. Every ℓ bytes are grouped into a portion for the remaining data of ***T***. Each portion is encrypted by simply exclusive-ORing itself with its precedence. The procedure is as follows. The first *d* bytes of the compressed video data ($v_1v_2...v_d$) that are not involved in the puzzling procedure are exclusive-ORed with *d* bytes of key stream ***S***, which are $s_{\ell+1}$ $s_{\ell+2...}$ $s_{\ell+d}$. The first ℓ ($\ell<L$) bytes ($t_1t_2...t_\ell$) of ***T*** are exclusive-ORed with first ℓ bytes (i.e. $s_1s_2\ldots s_\ell$) of the key stream ***S***. The key stream ***S***, which was generated in the puzzling step, is reused here in order to make the algorithm more efficient. After that the first ℓ bytes of ***T*** are used as key stream and exclusive-ORed with the second ℓ bytes of ***T***. Then the second ℓ bytes of ***T*** are exclusive-ORed with the third ℓ bytes of ***T*** and so on until the end of the frame is reached. As output we receive the *L* bytes long cipher text ***C*** ($c_1c_2...t_L$). Figure 7.7 shows the principle. Note that the frame header remains unencrypted, because it usually contains standard information.

Input text	v_1 v_2 ...v_d	t_1 t_2 … t_ℓ	$t_{\ell+1}$ $t_{\ell+2}$ … $t_{2\ell}$	$t_{2\ell+1}$ $t_{2\ell+2}$ … $t_{3\ell}$	… …t_{L-d}
$\oplus$					
Key stream	$s_{\ell+1}s_{\ell+2}\ldots s_{\ell+d}$	s_1 s_2 … s_ℓ	t_1 t_2 … t_ℓ	$t_{\ell+1}$ $t_{\ell+2}$ … $t_{2\ell}$	… …$t_{L-d-\ell}$
Cipher text	c_1 c_2 …c_d	c_{d+1} c_{d+2}…$c_{d+\ell}$	$c_{d+\ell+1}$ $c_{d+\ell+2}$…$c_{d+2\ell}$	$c_{d+2\ell+1}$ $c_{d+2\ell+2}$… $c_{d+3\ell}$	… … c_L

Note: v_i, s_i, c_i and t_i denote a data byte. The input text contains the temporary cipher text *T* and the first *d* bytes of the compressed video data.

Figure 7.7: Obscuring algorithm

7.3.3 Encoding procedure

The components of the *Puzzle* encoding procedure described above are summarized in Figure 7.8.

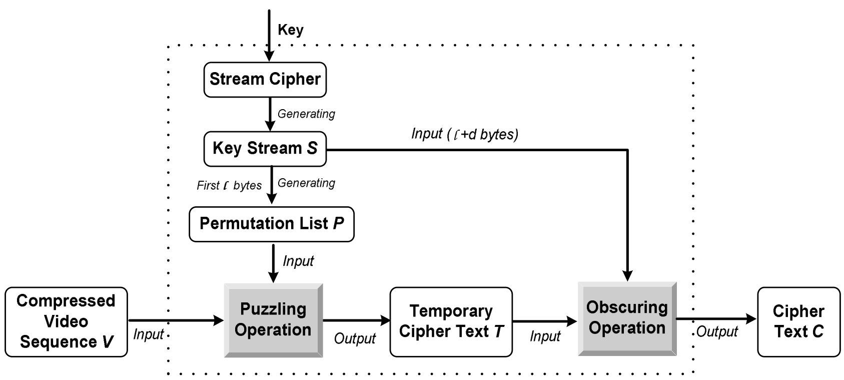


Figure 7.8: Encoding procedure

The puzzling operation partitions the incoming compressed video sequence and disorders it in accordance to the permutation list generated from the key stream (analogous to the picture disordering in the game). The outcome of this operation is the temporary cipher text. The obscuring operation encrypts a small part of the temporary cipher text with the key stream ***S***, while the rest is exclusive-ORed among itself (analogous to the picture turning over in the game). The output is the cipher text ***C*** which can be now securely delivered to its destination over the open network.

7.3.4 Decoding procedure

The original compressed video sequence can be re-established at the receiver's side by performing the encipher operations in reverse order (see Figure 7.9). First the cipher text is processed by the inverse obscuring operation, in which the first part of cipher text is exclusive-ORed with the key stream ***S*** to get the first part of the temporary cipher text. Then this part of temporary cipher text is used as a key stream to exclusive-OR it with the second part of the cipher text to obtain the second part of the temporary cipher text. The recovered second part of temporary cipher text keeps to be used as key stream to exclusive-OR it with the third part of the cipher text. This process is repeated till the end of the cipher text. As result, the entire temporary cipher text is recovered. This operation is much like the turning over the puzzled picture pieces from the reverse side to the right side. The puzzling operation applies the same permutation list as in the encoder to recover the original compressed video sequence from the temporary cipher text. This is based on the simple principle that a sequence will return to its original state after bytes in the sequence are exchanged two times by applying the same permutation list. The puzzling operation corresponds to the restoring of the original order of the picture pieces in the puzzle game.

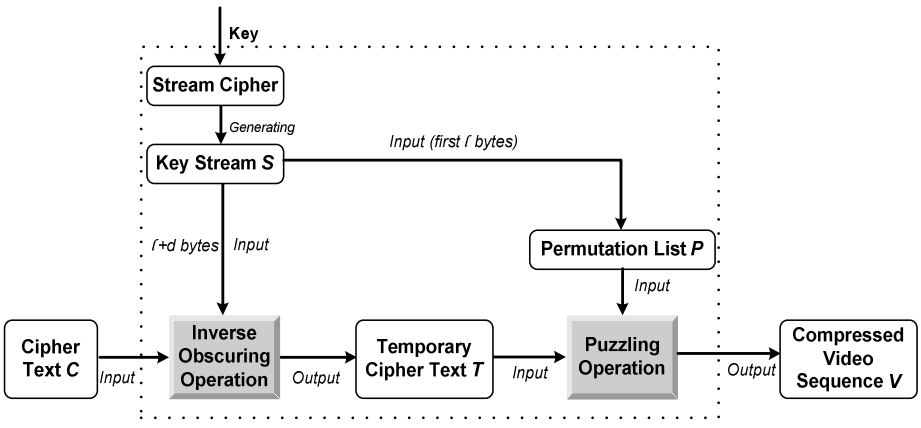


Figure 7.9: Decoding procedure

7.4 Evaluation of experimental results

This section describes experimental results of the application of *Puzzle* in encrypting video streams in comparison with the standard cipher AES. An analysis about its performance is given.

7.4.1 Determination of the parameter values in *Puzzle*

Before implementing the *Puzzle* algorithm, we have to determine the values of certain parameters used in the algorithm. These parameters control the different aspects of the *Puzzle* algorithm. In the obscuring operation the first ℓ bytes of the temporary cipher text $\boldsymbol{T}$ are scrambled with a stream cipher. The variable ℓ mainly determines the efficiency of the algorithm. If ℓ is equal to the length of $\boldsymbol{T}$, *Puzzle* becomes a stream cipher and no gain in the encryption speed would be achieved compared to a stream cipher. Therefore a value should be assigned which is as small as possible so that more encryption load can be reduced. On the other hand, a too small value of ℓ is not useful, because the ℓ bytes of the key stream $\boldsymbol{S}$ generated by a stream cipher are used not only for the encryption of the first ℓ bytes of $\boldsymbol{T}$ but also are used for the generation of the permutation list $\boldsymbol{P}$ (see Section 7.3.2). We set ℓ to 128 considering that the length of the permutation list $\boldsymbol{P}$ is not less than 64.

The compressed video data $\boldsymbol{V}'$ in the puzzling operation is split into at least *mb* blocks. The value of *mb* heavily influences the security of *Puzzle*. It is obvious that larger *mb* result in better security since the compressed video data $\boldsymbol{V}'$ is partitioned into more blocks. However, the larger value of *mb*, the heavier the computation load. As discussed in Section 7.3.2, the determination of *mb* should be a trade-off between security and efficiency. The value of *mb* should be confined in the range from 128 to 256. There are 64! permutations when *mb* is 128. This provides sufficient security because 64! is a big number. On the other side, *mb* is not allowed to exceed 256. Otherwise, the permutation list $\boldsymbol{P}$ may have a constant ending [175]. This is because we generate the permutation list $\boldsymbol{P}$ directly from the key stream $\boldsymbol{S}(s_1 s_2 \ldots s_\ell)$, where s_i denotes one byte whose value is less than 256. Therefore, we chose 128 for *mb* by balancing security and efficiency.

Finally, we have to make a decision which kind of stream cipher is applied for the generation of the $\ell+d$ bytes of the key stream. As there is no stream cipher that has emerged as a de facto standard so far, a standard block cipher (e.g. AES) is usually used as a stream cipher under some operation modes of a block cipher (e.g. CFB mode, OFB mode, CTR mode) [169]. We choose AES cipher working with CTR mode (i.e. AES-CTR) as the stream cipher in our algorithm. The attractive feature of the CTR mode is that a plaintext block can be recovered at the receiver side independently from the other plaintext blocks if the corresponding counter can be determined [168]. Thus it is inherently tolerant to packet

loss and re-ordering. This is very useful for the transmission of video data using UDP protocol over the Internet, where packet loss and re-ordering frequently occur.

7.4.2 Experimental results

We conducted a series of experiments for compressed video streams (MPEG-1, H.261) to measure the encryption speed of the *Puzzle* algorithm in comparison with the conventional cipher AES. All encryption speed tests were run on the platform SUN Ultra 10, while all encryption effect tests are carried out on the platform Windows 2000. One example is given for each video format.

MPEG-1

We exploited a demo MPEG clip *Table tennis* (352x240 pixels, see Figure 7.10 (a)) from the Berkeley University [170] to compare the performance of our algorithm for a MPEG-1 video stream. We make use of the *mplayer* [171] to play back the encrypted video stream. The outstanding feature of *mplayer* is that it can still play back a video stream that contains the syntax errors. The effects of encrypting *Table tennis* with *Puzzle* and AES are shown in Figure 7.10 (b) and (c) respectively. It demonstrated that no original video information is leaked out for both *Puzzle* and AES algorithms.

(a) *Table tennis* (b) Encrypted *Table tennis* using *Puzzle* (c) Encrypted *Table tennis* using AES

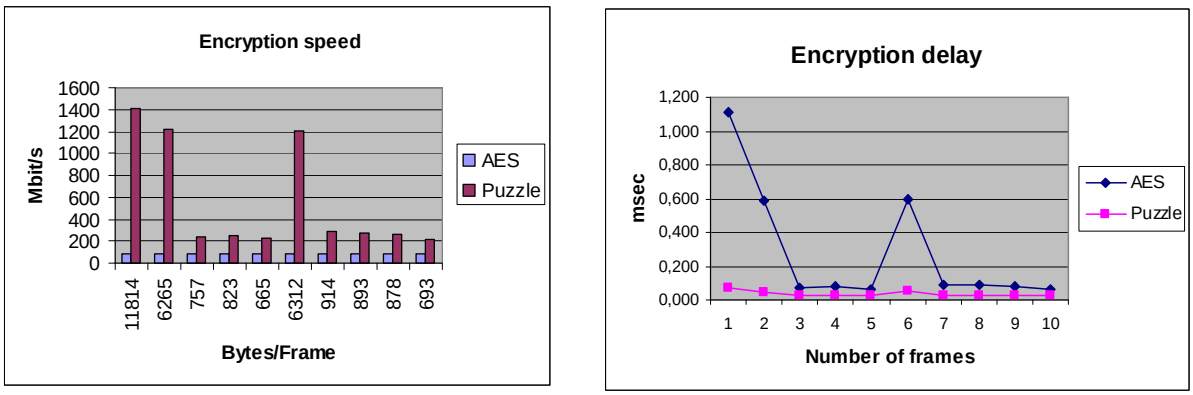
Figure 7.10: *Table tennis* MPEG-1 video and its encrypted effects

The *Table tennis* MPEG-1 video clip consists of totally 10 frames: one I-frame, two P-frames and seven B-frames. The video frame sizes vary, since MPEG-1 applies interframe coding. The size of each frame, the corresponding frame type, and the performance test result are summarized in Table 7.1 as well as in Figure 7.11(a). It shows that *Puzzle* is much faster than AES, especially for larger size frames. On average our algorithm accelerates the encryption speed about six times compared to AES in this *Table tennis* example.

Table 7.1: Performance comparison for MPEG-1 video clip *Table tennis*

Frame type	I	P	B	B	B	P	B	B	B	B	Average speed
Frame size (Byte)	11814	6265	757	823	665	6312	914	893	878	693	
AES (Mbit/s)	85	86	83	84	86	85	84	84	85	84	84.6
Puzzle (Mbits/s)	1406	1222	242	253	221	1202	281	274	259	220	558

In general, QoS guarantee is an appealing requirement for supporting real-time video applications (e.g. video conferencing). Delay and delay variance (jitter) are two crucial QoS parameters to be concerned which should be as minimal as possible. From Figure 7.11(b), we can see that the delay introduced by *Puzzle* is smaller than that of AES. Nearly no jitter is incurred by our algorithm. In contrast, a large jitter can be observed for AES. This is simply because the encryption speed of *Puzzle* varies with the frame size, i.e. the larger the frame size, the faster the encryption speed, whereas the encryption speed of AES nearly keeps unchanged for different frame sizes.

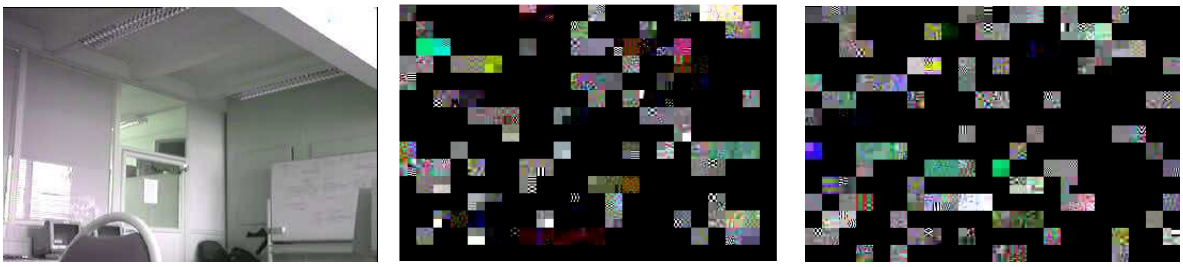


(a) (b)

Figure 7.11: Performance comparison between AES and *Puzzle* for the MPEG-1 video clip *Table tennis*

H.261

H.261 video streams are rarely available in the open source. We measured the encryption speed of our algorithm using an H.261 video stream *Lab* of our own computer lab (see Figure 7.12(a)) which is generated by the SunVideo Plus subsystem [172] installed in the workstation Sun Ultra 10. The SunVideo Plus card can make use of the onboard video processor to compress the raw video captured by the video camera into diverse compressed video formats (including H.261) in real-time. This is particularly useful for interactive multimedia applications, e.g. video conferencing. Figure 7.12 (b) and (c) illustrate the encryption effects of this *Lab* video stream for the *Puzzle* and AES algorithm respectively.



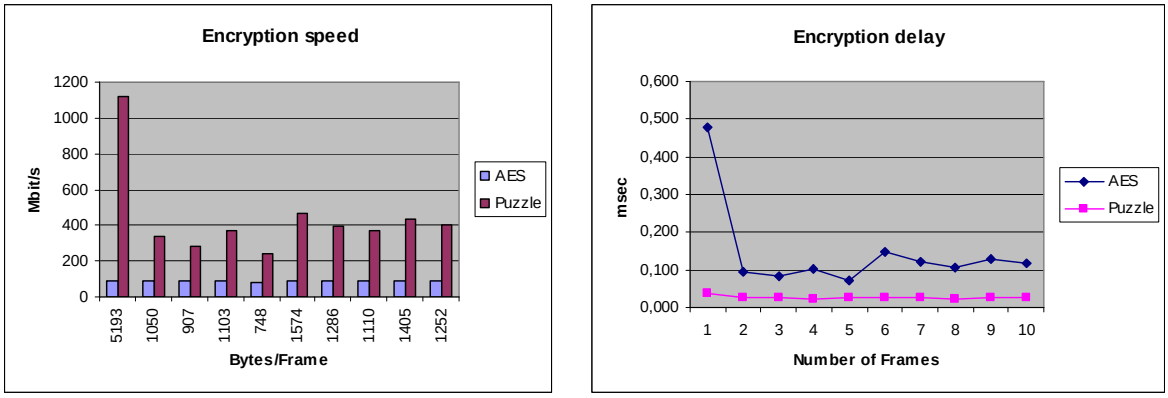
(a) *Lab* (b) Encrypted *Lab* using *puzzle* (c) Encrypted *Lab* using AES

Figure 7.12: *Lab* H.261 video and its encrypted effects

The *Lab* H.261 video stream contains 10 frames: one I-frame and nine P-frames. Its resolution is CIF (352x288 pixels). The performance comparison between the *Puzzle* algorithm and AES is depicted in Table 7.2 and Figure 7.13. We can draw the similar conclusions as above from these test results for the H.261 video stream.

Table 7.2: Performance comparison for H.261 video stream *Lab*

Frame type	I	P	P	P	P	P	P	P	P	P	Average speed
Frame size (byte)	5193	1050	907	1103	748	1574	1286	1110	1405	1252	
AES (Mbit/s)	87	87	85	86	84	86	86	85	86	86	85.8
Puzzle (Mbits/s)	1123	336	279	368	239	466	396	370	432	401	441



(a) (b)

Figure 7.13: Performance comparison between AES and *Puzzle* for the H.261 video stream *Lab*

7.4.3 Performance analyses

The computation cost of *Puzzle* comprises three parts: the ℓ+d bytes (i.e.128+d) encryption with AES-CTR, the permutation list ***P*** generation and the L-d bytes exclusive-OR operation

and exchange. After running the empirical tests, we have acquired that the computation complexity of the permutation list ***P*** generation is approximately equivalent to a 92 bytes encryption with AES, and one byte exclusive-OR operation and exchange corresponds to 1/20 computation load of one byte encryption with AES. Thus the encryption speed gain of our algorithm compared to AES can be expressed by the following formula.

$$Gain = \frac{T_{AES}(L)}{T_{AES}(128+d)+T_{PM}+T_{EXOR}(L-d)} = \frac{T_{AES}(L)}{T_{AES}(220+d)+\frac{1}{20}T_{AES}(L-d)}$$
$$= \frac{T_{AES}(L)}{T_{AES}(220+\frac{19}{20}d+\frac{1}{20}L)} \qquad (7.4)$$

where $T_{AES}\ (L)$ denotes the time spent in encrypting L bytes with AES-CTR cipher, T_{PM} stands for the time to generate the permutation list ***P***, $T_{EXOR}\ (L)$ means how long are needed to implement L bytes exclusive-OR operation and exchange.

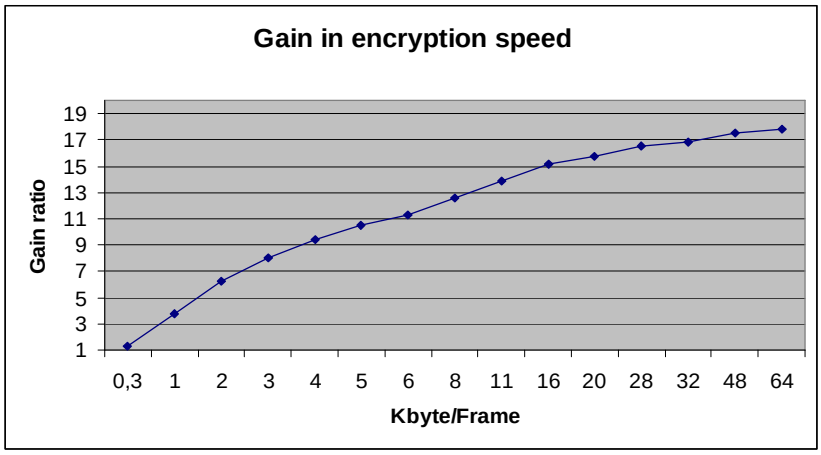


Figure 7.14: Gain in encryption speed of *Puzzle* compared to AES

The detailed calculation results are depicted in Figure 7.14. They show that the encryption speed gain of *Puzzle* almost linearly ascends with the increment of the frame size till the size of 16 Kbyte. After that point the gain is slightly improved. The gain reaches its highest point at 64 Kbyte frame size with about 17 and it arrives its lowest point at 300 byte frame size with 1. This indicates that *Puzzle* is well suitable for high resolution video streams which usually have a large frame size.

7.5 Security analyses of the *Puzzle* algorithm

In this section we evaluate the security properties of the *Puzzle* algorithm. A good crypto system should withstand the following most important attack classes [70], [173], [174]:

- *Cipher text-only attack*: No additional information is available for an adversary except for the cipher text. The main goal of the adversary is to recover the plaintext from the cipher text as much as possible.
- *Chosen-plaintext attack*: The attacker can choose the plaintext as he/she needs and insert it into the encryption system. Consequently he/she can obtain the corresponding cipher text. The goal of the attacker is to deduce the key under these plaintext/ciphertext pairs so that he can decrypt any messages encrypted with that key.
- *Known-plaintext attack*: The attacker does not only know the cipher text but also some plaintext for several messages. He/she aims at deducing the key or recovering more plaintext on the basis of the known plaintext/cipher text pairs.
- *Differential attack:* The attacker attempts to reconstruct the encryption key by studying the differences between the plaintext and the respective ciphertext pairs. The differential cryptanalysis can reduce the complexity of attacking a cipher by half. Generally speaking, it is a specific variant of a *chosen-plaintext attack*.

The security analyses of our algorithm are given for each class of attack as follows.

Ciphertext-only attacks

Based on the cipher text an adversary has two possibilities for trying to re-establish the original frame encrypted with *Puzzle* (see Figure 7.9). He/she can either attempt to break the puzzling and then the obscuring operation or to crack these steps in the reverse order. The first attack corresponds to the situation when the child first tries to put the disordered pieces to their correct position only looking at their backside and then turns the whole correctly reordered picture to the right side. In *Puzzle* each frame is split in at least 128 blocks in the puzzling operation, i.e. more than $64! = 1.27 \times 10^{89}$ trials to reconstruct a single original frame. This is obviously computationally infeasible to be broken, especially as a 128 bit key length standard block cipher is believed to be computationally secure enough today. The number of possible permutation for such standard block cipher is, however, only $2^{128} = 3.4 \times 10^{38}$.

The second attack resembles the situation when the child first turns the disordered pieces to the right side one by one and then re-establishes the picture using content information. In *Puzzle* the cipher text $\boldsymbol{C}$ is generated by connecting two puzzled plaintexts fragments using the exclusive OR operation except the first $\ell+d$ bytes of the obscuring operation. As shown in [166] the computation complexity to obtain 10 bytes MPEG compressed video sequence

by separating two 5 bytes long exclusive-ORed plaintext is equivalent to that of breaking a 64-bit key length block cipher which has 2^{64} combinations. As mentioned in Section 7.4.3, we recommend to applying *Puzzle* on video sequences with a frame length larger than 300 bytes. Accordingly the attacker has at least to try all $30x2^{64} \approx 2^{69}$ combinations to obtain the plaintexts of the disordered blocks for a single frame.

Chosen-plaintext attacks

An attacker encrypts some compressed video frames of his/her choice with the *Puzzle* algorithm, and obtains the respective ciphertexts. As show in Figure 7.7 and 7.9, it is not difficult for an attacker to recover these key streams ***S*** with the knowledge of these plaintext/ciphertext pairs. However, it is impossible to derive the encryption key ***K*** from a known key stream ***S,*** since AES-CTR is a confidentiality mode [168] which is secure against chosen-plaintext attacks. Therefore our algorithm can withstand these attacks.

Known-plaintext attacks

An attacker is unable to deduce the encryption key ***K*** from his/her known plaintexts and ciphertexts. The reason is the same as the above explanations to the chosen-plaintext attacks. Now we discus to what extend our algorithm prevents an attacker obtaining further plaintexts with some known plaintexts by considering the following two cases:

- *Case 1:* An attacker attempts to reconstruct more frames when he/she knows an entire compressed video frame. It is impossible for an attacker to achieve this goal when he/her wants to crack our algorithm because we encrypt each frame with a distinct key stream ***S***.

- *Case 2:* An attacker tries to recover an entire frame from the corresponding ciphertexts when he/she knows a partial plaintext of that frame. As analyzed in [175], our algorithm suffers from this attack when it is used for VoD services where some information may be constantly showed in the screen, such as copyright declaration. However, this attack is only effective to I-frames whose respective plaintexts (compressed bytes) are partially known to the attacker a prior, since B-and P-frames will not contain this constant information any more due to using motion compensated prediction technology. The compromise of one I-frame will not lead to the compromise of other frames because each frame is separately encrypted. A typical encoding group of pictures of MPEG frames is IPBBBPBBBPBBB. These 12 frames have only one I-frame. An attacker may recover one I-frame for every 12 frames if it knows the part information of that I frame, whereas the other P-and B-frames are not affected. This means that an attacker could get two frames of every 25 frames MPEG video sequences (frame rate: 25 frames/second). It makes no great sense for an attacker when

he/she can recover only small amount of frames. It will be a "quick" movie when he plays these inconsecutive recovered frames. Considering that the final result of attacks is a useless MPEG video sequence (inconsecutive), it is therefore deeply convinced that an attacker would prefer to buy the VOD services directly rather than to spend time and money to launch such attacks. Such attacks are ineffective to our algorithm when it is applied to video conferences because an attacker has no way in advance to know which information will constantly appear in the conference.

In fact, most existing video encryption algorithms only take the first case into account. So does the current version of *Puzzle* algorithm. This is because attacks of second case do not substantially affect their practical applicability. Certainly, such a small cryptographic flaw should be eliminated for a perfect cipher. One possible approach to solve this problem is to add one more puzzling step after the obscuring step to resist against attacks mentioned in case 2. The security and performance of this improved *Puzzle* algorithm need further studies.

Differential attacks

They are generally viewed a specific kind of *chosen-plaintext attacks*. As discussed above, such attack is not effective to our scheme. On the other hand, attackers might apply the basic idea of differential cryptanalysis to launch a *specific* ciphertext-only attack by analyzing the ciphertext of our scheme without the knowledge of the respective plaintext for the specific structure of our scheme. The order of encryption procedure in our scheme decides, whether our scheme is strong enough to withstand such a specific ciphertext-only attack. In [109] we first obscured the original frame and then puzzled the obscured one. This encryption order is suspect to be too weak for specific ciphertext-only attacks, because the edges values of the blocks of the original video frame tend to be very close. These close values are inherited to the obscured frame in this encryption order. The attacker might determine which blocks might be neighbors using this information. For that reason, we have changed the encryption order, i.e. first puzzling then obscuring. The edges of neighbor blocks will now have significantly different values so that such an attack is avoided.

To sum up it can be said that our scheme can be viewed as a 69 bit key length block cipher. Recently a 64 bit key length block cipher has been reported to be broken using brute-force attacks [176]. It, however, took five years to complete this attack. Thus the security level of the *Puzzle* algorithm is sufficient enough for the commonly used multimedia applications such as VoDs or video conferences.

7.6 Summary

Video encryption has been a hot research topic for the past decades due to its potentially wide market prospect. The emerged approaches do not pose an appropriate balance between security and efficiency. Most of them rely on an integration of the encryption with the compression into one step. This makes them unable or not harmony for inclusion into existing multimedia systems.

In this chapter we presented the video encryption algorithm *Puzzle* for encrypting video communication in real-time. As a compression-independent encryption algorithm, *Puzzle* inherently makes no impairment on the compression efficiency and is easily to integrate into available multimedia applications. *Puzzle* achieves a sufficiently fast encryption speed to meet the real-time requirements of mostly used multimedia applications, especially for high resolution video streams. Moreover, it withstands most important types of attacks. In other words, it provides a good trade-off between security demands and encryption efficiency. These outstanding features of the *Puzzle* algorithm are really appealed by a video encryption algorithm. Thus it is anticipated that our algorithm likely finds its wide applications in multimedia systems in the near future.

Chapter 8

Final Remarks

This chapter summarizes the main contributions of this dissertation and outlines some issues for future research.

8.1 Contributions

Although the client/server model is still dominant in the Internet, many applications increasingly apply the newly emerging peer-to-peer technology. The speed of this paradigm shift has an accelerating tendency since the incentive of the popular music sharing application Napster. The P2P networks are distributed systems in which peers share resources by direct data exchange without requiring a centralized server or authority. It possesses a number of important benefits compared to the traditional centralized client/server model: fault-tolerance, scalability, cost-effectiveness and others. It is very likely that the P2P model will obtain a greater importance in providing next-generation Internet services.

A variety of applications have already adopted the P2P model to offer services. Typical applications are distributed computing, file sharing, and communication/collaboration. Instant messaging is a successful example for the latter. Video conference systems, as a typical real-time interactive collaborative application, however, rarely follow the P2P principle yet. Most video conference systems still rely on a centralized conference server to hold meetings like the H.323 systems. Only a few P2P video conference systems have emerged. The BRAVIS system is one of the earliest ones among them. The application of the P2P model in designing video conferencing systems brings additional benefits to its users besides the aforementioned. The inherently ad-hoc nature of the P2P model enables people to spontaneously set up meetings without relying on a centralized server and without having to be bound to a service provider. This makes a conference system closer to the nature of social behaviours in daily face-to-face meetings. Thus it can be expected that more video conference systems will apply the P2P model. However, such technological changes bring a number of technical challenges to the system design. Security is one of the most critical

ones that a P2P video conference system has to solve. Standards, such as ITU-T H.235, have specified a secure framework for video conference systems. However, they are dedicated to client/server-based systems (e.g. H.323 systems) and are thus not applicable to P2P systems. No standard exists for specifying a secure architecture used for a P2P video conference system. This thesis has pursued the aim to close this gap by proposing an appropriate security architecture and showing how it can be incorporated into a P2P conference system using our video conference system BRAVIS as example.

As a first step towards designing a secure P2P video conferencing system, we have analyzed which kinds of threats a P2P conferencing system is really confronted with by using the effective STRIDE model. To mitigate the possible attacks a P2P conferencing system has to support the following security goals: *confidentiality*, *data integrity*, *user authentication*, and *authorization*. User authentication forms the basis of the other three goals. Without an appropriate and rigorous authentication measure an attacker may join a meeting and learn the content of the meeting even if other security services are in place. Identity certification is one of commonly used solutions used for user authentication. We surveyed and compared the advantages and disadvantages of two kinds of certification schemes used in P2P systems: centralized certification (like PKI) and decentralized certification (like PGP). We argued that the centralized certification scheme is the unique choice for a P2P system used in serious environments (e.g. an international enterprise) to achieve the reliable user authentication.

Nowadays virtual private networks (VPNs) are broadly employed for securing the communication across public networks. There are four kinds of possible VPNs in a TCP/IP protocol stack: data link layer VPNs, IPsec VPNs, SSL VPNs, and application layer VPNs. We explored the feasibility of their direct deployment for securing a P2P conference in terms of end-to-end security, group key management, and flexible security policy enforcement and found that lower layer VPNs do not well fulfill these requirements. So a security architecture especially designed for a P2P conference system (i.e. an application layer VPN) is the most appropriate solution to meet the stringent security and efficiency requirements. As example for such a security architecture, we presented the security solution applied to the P2P video conference system BRAVIS. The security architecture for a P2P conferencing system is composed of a set of reusable building blocks: security policy management, authorization, decentralized group key management, and data security. The last two modules were the primary concerns of this thesis, since they definitely decide the success of a secure P2P conferencing system.

Designing a decentralized group key management protocol used for a P2P conferencing is usually viewed as a challenging task. Such protocol has to meet a couple of rigid security requirements for its use like key authentication, forward and backward confidentiality, collusion freedom, and others. In addition, its employment should not incur great performance degradation to the system because interactive real-time communication is the key feature of a P2P conferencing system which has to be preserved when introducing a group key management protocol. This implies that the proposed protocol should be not only secure but also efficient enough (i.e. minimal rekeying delay). Although several such protocols are available, none of them completely fulfills the desired security demands during a key renewal period. Therefore we have proposed a novel decentralized group key exchange protocol, VTKD. It consists of two parts: a mutual authentication of the partners and a secure key renewal. The protocol uses a virtual token to determine the partner responsible for key generation and distribution. VTKD fully fulfills the relevant security demands concerning group key exchange. Moreover, it is more efficient related to key renewal delay than existing key exchange protocols in a small group setting because it needs only one communication round and uses mostly symmetric crypto operations during a key renewal procedure. Therefore the VTKD protocol is more suitable to be applied in a P2P conferencing system than other existing protocols.

Special concerns have to be paid to preserve confidentiality of video data. It is usually difficult for a standard algorithm which aims at encrypting text data to encrypt video data in real-time due to the huge data volume. Accordingly a specific algorithm is strongly demanded to encrypting the video data. Exploring several available algorithms it has shown that they possess various shortcomings when applied in a P2P conferencing system. Some of them provide a significant imbalance between security and efficiency; others severely degrade the compression efficiency of the video encoder. Some of them are not easy (or impossible) to incorporate into existing multimedia systems. Inspired by the children game Puzzle, we proposed a novel compression-independent video encryption algorithm, called *Puzzle*. It overcomes the shortcomings of existing video encryption algorithms. The essential features of are: easy to integrate into existing multimedia system, no impairment on compression efficiency, and good trade-off between security and efficiency. The further outstanding advantage of this scheme is the drastic reduction of encryption overhead for high resolution video. Thus it is most likely that the use of the *Puzzle* algorithm is not limited to P2P conferencing systems, but spreads to other kinds of multimedia systems, such as Video on Demand.

8.2 Outlook

As long as P2P technologies are not matured, many fundamental issues are still open. Security is one of the most important ones among these issues. Basically the considerably mature security architectures which have been broadly employed in client/server systems are not suitable for P2P systems because of the significant differences between these two kinds of systems. This is in particular true for real-time P2P systems. Consequently, we have argued for designing specific secure architectures for P2P conferencing systems. In this context, we have proposed a secure and efficient decentralized group management protocol and a novel video encryption algorithm to meet the stringent real-time constraints. Closely related to the currently accomplished work, the following two directions are considered worth to be explored for future work:

- *Completing the VTKD protocol*: The VTKD protocol in its present form supports only the group operations *joining* and *leaving*. It does not consider group partitioning and merging. When a group is partitioned group communication simply terminates. This rule was taken to meet practical demands of video conferences usually requiring that all participants and not only a part of them have to be present to discuss a dedicated topic. Moreover, it is basically unexpected when partitioned groups will be merged again because their partitioning was mainly caused by a network failure. In short, introducing the group partitioning and merging mechanisms into a real-time person-to-person group communication makes no great sense[3]. Certainly it is useful for a machine-to-machine group communication in which no people are involved, such as a group data replication system in which critical data are automatically delivered to several different places. To adapt this kind of applications, the group partitioning and merging mechanisms should be introduced into the VTKD protocol. How to add these two mechanisms in VTKD needs further studies.

- *Video/audio data integrity*: In the thesis we have addressed the confidentiality of video data with a novel video encryption algorithm to meet real-time requirements. Basically special considerations are also needed for verifying the integrity of video/audio data. Video/audio data are transmitted over the Internet using the UDP protocol which has no retransmission mechanisms when bit errors are detected. Receivers are not able to determine whether these errors originate from (unavoidable) transmission errors or from deliberate malicious attacks using traditional integrity check techniques (e.g. MAC codes). Some research efforts have been devoted to

[3)] Of course, it is convenient for users in some cases to actively partition the group into some subgroups and later merge them again for the discussions concerning different topics. However, people usually hold a video conference only for a certain purpose. It seldom serves several purposes. If this is really needed, people can set up separate conferences.

addressing this issue. But most of them rely on watermarking technologies which are computationally intensive. So they are basically not applicable to real-time applications such as video conferencing. A novel and efficient integrity verification algorithm used for video/audio data is highly desired in this sense.

Generally speaking, security is difficult to achieve in a P2P system due to its decentralized nature. This difficulty makes many general security issues still not well addressed in the P2P security research area. They play decisive roles in the operation of a P2P network. At least the following two issues are deserved to pay more attention:

- *Secure routing*: In the traditional Internet, the routing tables in the routers are centrally controlled by the Internet service providers (ISPs). So attackers have less chance to alter these routing tables. The situation in the P2P networks is significantly different. The routing tables are managed by peers themselves, and no central authority is in charge of the administration of these tables. As a result, two kinds of attacks may be launched to destroy the normal operation of a P2P network: (1) an attacker compromises a peer and modifies the routing table of that peer (2) a peer itself is a malicious node which deliberately generates an error routing table. Thus it is highly desired to design a secure P2P routing protocol for the defence against these attacks.

- *Reputation management*: It is used to determine the extent of trustiness of a user in the system by evaluating his/her previous behaviours. This is an important requirement for a system to be used for the electronic commerce. In the client/server systems, reputation management is easy to realise since there is single entity responsible for the maintenance and distribution of reputation information (e.g. reputation management in eBay). The decentralised nature of the P2P systems makes the reputation management quite difficult. The reputation information is scattered into the entire network. How to ensure that the reputation information is not modified by an attacker or malicious peer is still an open issue.

Appendix A

Benchmarks of Crypto Operations

The speed benchmarks for the cryptographic algorithms used in this thesis are listed in the following table. These results are measured on a Dell computer equipped with Intel Xeon 2.6 GHz processor under Linux using OpenSSL's cryptographic library.

Algorithms	**Symmetric crypto operations**			**Asymmetric crypto operations**		
	HMAC (SHA-1)	AES (128-bit key)		RSA 1024 Signature	RSA 1024 Verification	DH 1024 key Agreement
		Encryption	Decryption			
Speed	25 Mbyte/s	50 Mbyte/s	50 Mbyte/s	5 ms/operation	0.16 ms/operation	12.5 ms/operation

Appendix B

State Diagram of VTKD

VTKD is based on notifications of membership changes indicated by the group management module to trigger the group key refreshment. Whenever the group composition changes (including the leave or crash of the token holder) the underlying group communication protocol indicates the related service primitives to the group management module. The group management module passes theses service primitives to VTKD to trigger the group key renewal. When a group member failed to update the group key in VTKD, it has to leave the group and indicates this to the group management module to update the group composition via the underlying group communication module. In the meanwhile, this invokes the group key refreshment. Figure B1 shows the state diagram of VTKD. In Table B1 some service primitives are listed, which we assume for the interaction between VTKD and the group management module.

Table B1: Service primitives of the group communication protocol

Service primitive	Meaning
JOINntf	A new member joined the group
LEAVErequ	A member requests to leave the group
LEAVEntf	A member left the group
LEAVEanc	A failed authentication member is forced to leave

Other events and actions used in the state diagram are described in Table B2.

Table B2: Meaning of symbols used in the state diagram of VTKD

Symbol	Meaning
M_{J1}-M_{J4}	Authentication messages exchanged between token holder and invitee
Authf	Failed authentication
M_{Jf}	Notification of failed authentication to the token holder
M_{J5}, M_{L1}	Group key renewal messages for join and leave, respectively
Rek	Rekeying
MACf-M_{J5}, MACf-M_{L1}	Failed message authentication of messages M_{J5} and M_{L1} respectively
$timeout_1$, $timeout_2$, $timeout_3$	Timeouts of timers supervising the authentication procedure
Init	Initializing the member process
Quit	Releasing the member process

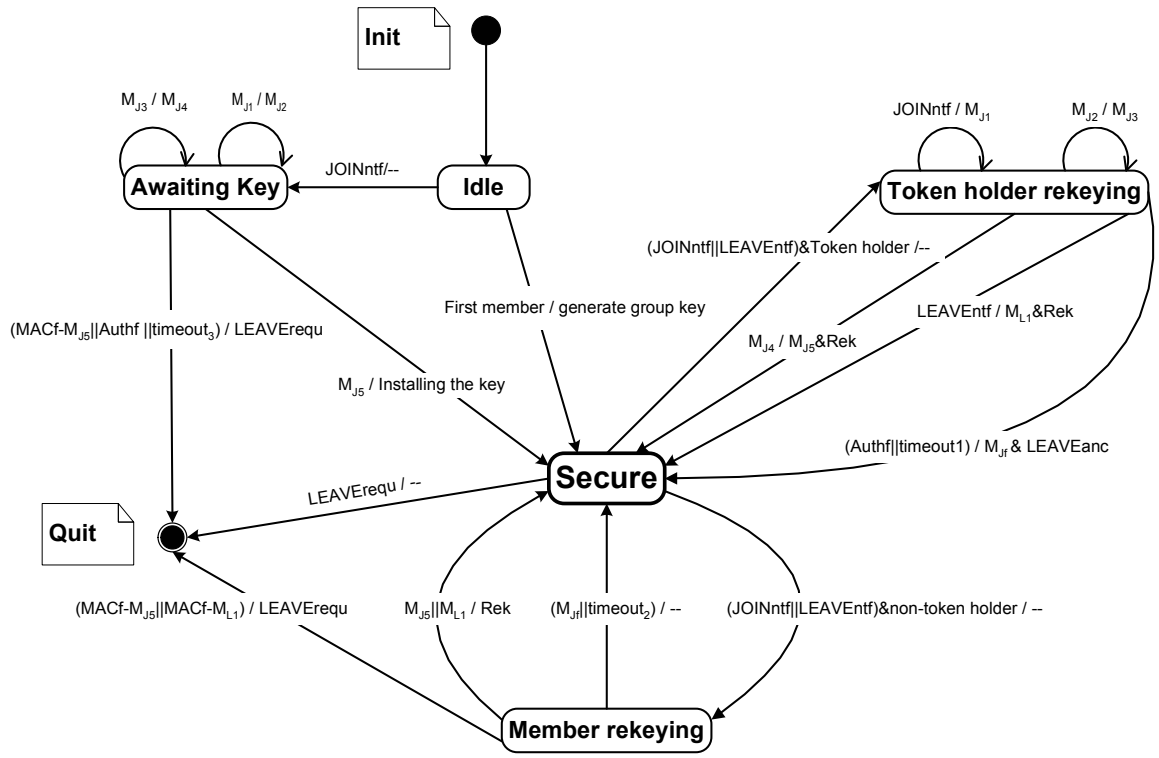


Figure B1: State diagram of VTKD

As shown in Figure B1 VTKD has six states with the following meaning:

- ***Idle***: Initial state of all group members.
  The first member generates the group key and changes to state ***Secure***. Later joining members switch from ***Idle*** state to ***Awaiting key*** state when a JOINntf is indicated.

- ***Secure***: All group members communicate securely using the same group key.
  When the group membership changes, which is indicated by JOINntf or LEAVEntf, the token holder enters the state ***Token holder rekeying***. Other members move to the state ***Member rekeying***. A member who received a LEAVErequ leaves the secure communication and releases the process.

- ***Token holder rekeying***: The token holder refreshes the group key.
  When a member joins the token holder first authenticates the invitee. The token holder will stay in the same state when it receives the events JOINntf and M_{J2}. After receiving message M_{J4} it generates the new key and forwards it with message M_{J5} to the group, and moves itself to state ***Secure***. If the token holder fails to authenticate the invitee or the time used for authentication is over ($timeout_1$) it informs the group

members with message M_{Jf} and LEAVEanc. The primitive LEAVEanc initiates the Forced-leave mechanism of the GCP protocol to exclude the failed authentication member from the group. The group key keeps unchanged. After that the token holder changes to state ***Secure***.

When a member leaves the token holder generates a new group key and multicasts it with message M_{L1} to the group. The token holder moves to ***Secure***.

- ***Member rekeying***: The non-token holders refresh the group key.
  When receiving message M_{J5} or M_{L1} from the token holder each member first proves the authenticity of M_{J5} or M_{L1}. If the message authentication is successful the member updates the group key and change to state ***Secure***. Otherwise an active attack has to be assumed. Since this is rather seldom in practice it was decided that the member has to leave the group sending a LEAVErequ to the group management module to notify the group about its leave. It has to be explicitly invited again.

  If a member receives message M_{Jf} or the waiting time for message M_{J5} or M_{L1} expired ($timeout_2$) it moves again to state ***Secure***. The group key keeps unchanged.

- ***Awaiting key***: The new member awaits the group key.
  First the new member authenticates the token holder via the received message M_{J3}. If the authentication failed or the time used for authentication ($timeout_3$) expired, the new member leaves the group sending a LEAVErequ to the group management to notify the group about its leave. When receiving message M_{J5} it proves the authenticity of the message. If the authenticity is given the new member installs the group key and transfers to state ***Secure***. Otherwise it stops the joining process sending a LEAVErequ.

Acronyms

ACL	Access Control List
ACM	Association for Computing Machinery
AES	Advanced Encryption Standard
AH	Authentication Header
ARPANET	Advanced Research Projects Agency Network
ATM	Asynchronous Transfer Mode
BER	Bit Error Rate
BRAVIS	BRAndenburg VIdeo conference System
CA	Certification Authority
CCS	Change Cipher Spec Protocol
CHAP	Challenge Handshake Authentication Protocol
CPU	Central Processing Unit
CRL	Certificate Revocation List
DCT	Discrete Cosine Transformation
DDOS	Distributed Denial-of-Service
DES	Data Encryption Standard
DH	Diffie-Hellman
DHT	Distributed Hash Table
DOS	Denial-of-Service
DTSS	Digital Time Synchronization Protocol
ESP	Encapsulating Security Payload
FTP	File Transfer Protocol
GCP	Group Communication Protocol
GUI	Graphic User Interface
GKMP	Group Key Management Protocol
HTTP	Hypertext Transfer Protocol

IDS	Intrusion Detect System
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IFIP	International Federation for Information Processing
IKE	Internet Key Exchange
IP	Internet Protocol
IPsec	Internet Protocol Security
ISP	Internet Service Provider
ISDN	Integrated Services Digital Network
IT	Information Technology
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
JPEG	Joint Photographic Experts Group
L2F	Layer 2 Forwarding
L2TP	Layer 2 Tunnelling Protocol
LAN	Local Area Network
LAC	L2TP Access Concentrator
LDAP	Lightweight Directory Access Protocol
LKH	Logical Key Hierarchy
LNS	L2TP Network Server
MAC	Message Authentication Code
MC	Multipoint Controller
MCU	Multipoint Control Unit
MP	Multipoint Processor
MPEG	Moving Picture Experts Group
MPLS	Multiprotocol Label Switching
NAS	Network Access Sever
NTP	Network Time Protocol
PAL	Phase Alternation Line
PC	Personal Computer
PCI	Peripheral Component Interconnect
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
PPTP	Point-to-Point Tunnelling Protocol

PSTN	Public Switched Telephone Network
P2P	Peer-to-Peer
QoS	Quality of Service
RA	Registration Authority
RBAC	Role Based Access Control
RFC	Request for Comments
RLC	Run Length Coding
RSA	Rivest-Shamir-Adelman
RTP	Real-time Transport Protocol
SA	Security Association
SAD	Security Associations Database
SARS	Severe Acute Respiratory Syndrome
SHA-1	Secure Hash Algorithm 1
SIP	Session initiation protocol
SRTP	Secure Real-time Transport Protocol
SPD	Security Policy Database
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TGDH	Tree based Group Diffie-Hellman
TKD	Token based Key Distribution
TTL	Time-to-Live
TTP	Trusted Third Party
TLS	Transport Layer Security
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
VEA	Video Encryption Algorithm
VLC	Variable Length Coding
VoD	Video on Demand
VPN	Virtual Private Network
VTKD	Virtual Token based Key Distribution
WAN	Wide Area Network
WWW	World Wide Web

References

[1] TANDBERG: PoperASW and TANDBERG International Survey Results Overwiew. http://www.tandberg.net/collateral/video_communication/roper_survey_fact_sheet.pdf, Nov. 2003.

[2] Wainhouse Research: The Business Case for Videoconferencing. Publication #567, March, 2002.

[3] eFinancialNews: Sars Boosts Video Conferencing. 12 May 2003. http://www.avistar.com/assets/docs/FinancialNews_0503.pdf

[4] Wainhouse Research: Volume 1: Audio, Video, and Web Conferencing Infrastructure Products. 2004. http://www.wainhouse.com/reports/WR_RMC04_V1_summary.pdf

[5] Wainhouse Research: Volume 2: Videoconferencing Clients. 2004. http://www.wainhouse.com/reports/WR_RMC04_V2_summary.pdf

[6] ITU-T: Recommendation H.320-Narrowband visual telephone systems and terminal Equipment. http://www.itu.int, May 1999.

[7] York telecom: ISDN to IP Videoconferencing Migration. http://www.yorktel.com/images/whitepapers/MigrationWhitePaper.PDF

[8] M. Zühlke: Distributed Organzied Multiparty Video Conferences for Closed Groups in the Internet. Ph.D. thesis, Brandenburg University of Technology Cottbus, 2004 (In German).

[9] Network Research Group of Lawrence Berkeley National Laboratory: vat - LBNL Audio Conferencing Tool. http://www-nrg.ee.lbl.gov/vat/.

[10] Networked Multimedia Research Group at University College London: Videoconferencing Tool. http://www-mice.cs.ucl.ac.uk/multimedia/software/vic/.

[11] GMD Fokus: USMInT - Universal Scalable Multimedia in the Internet. http://www.fokus.gmd.de/research/cc/glone/projects/usmint/.

[12] ITU-T: Recommendation H.323v5--Packet Based Multimedia Communication Systems. http://www.itu.int, July 2003.

[13] Polycom: Your Videoconferencing and Speaker Phone Source! Polycom Worldwide. http://www.polycom.com/.

[14] First Virtual Communications: New Click to Meet 4.1 is Here! Integrated Communication Solutions for Connecting Your Teams, http://www.fvc.com/eng/products/

[15] VCON Desktop Videoconferencing GmbH: VCON steht für Videoconferencing. http://www.vcon.de.

[16] Microsoft GmbH: Netmeeting. http://www.microsoft.de.
[17] The BRAVIS video conference system. http://www.bravis.tu-cottbus.de.
[18] DaViKo - Gesellschaft für digitale audiovisuelle Kommunikation mbH: daViKo. http://www.daviko.com.
[19] S. Kent and R. Atkinson: Security Architecture for the Internet Protocol. IETF RFC 2401, Nov. 1998.
[20] T. Dierks and C. Allen: The TLS Protocol Version 1.0. IETF RFC 2246, Jan.1999.
[21] J. Wroclawski: The Use of RSVP with IETF Integrated Services. IETF RFC 2210, Sept. 1997.
[22] S. Blake, D. Black, M. Carlson, E.Davies, Z. Wang and W. Weiss: An Architecture for Differentiated Services. IETF RFC 2475, Dec. 1998.
[23] ITU-T: Recommendation H.235v3--Security and Encryption for H-series (H.323 and other H.245-based) Multimedia Terminals. May 2003.
[24] B. G. Haskell, A. Puri, and A. N. Netravali: Digital Video: An Introduction to MPEG-2. Kluwer Academic Publishers, 1996.
[25] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson: RTP: A Transport Protocol for Real-Time Applications. RFC 3550, July 2003.
[26] ITU-T: Recommendation H.245v9--Control Protocol for Multimedia Communication. Oct. 2002.
[27] The International engineering consortium: H.323, Web Proforum Tutorials. http://www.iec.org
[28] The Napster home page, www.napster.com.
[29] IRTF Research Groups, Peer-to-Peer Research Group, http://www.irtf.org/charters/p2prg.html
[30] D. S. Milojicic, V. Kalogerali, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, z. c. Xu: Peer-to-Peer Computing. HP white paper HPL-2002-57, Mar. 2002.
[31] M. Roussopoulos, M. Baker, D. S. H. Rosenthal: 2P2P or Not P2P? http://arxiv.org/abs/cs.NI/0311017.
[32] SETI@HOME: http://www.setiathome.ssl.berkeley.edu/
[33] AOL Instant Messager: http://site.aol.com/aim/about.html
[34] Groove: http://www.groove.net.
[35] The Gnutella forums home page, http://www.gnutelliums.com/.
[36] A. Rowstron and P. Druschel: Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platform, Nov. 2001.
[37] I. Stoica, R. Morris, D. Karger, F. Kaashoek and H. Balakrishnan: Chord: A Scalable Peer-to-Peer lookup Protocol for Internet Applications. IEEE/ACM Transactions on Networking, 11(1):17-32, 2003.

[38] S. Ratnasamy, P. Francis, M. Handley, R.Karp, and S. Shenker: A Scalable Content-Addressable Network. In Proc. ACM SIGCOMM. Aug. 2001, pp. 161-172.

[39] M. Datar: Butterflies and Peer-to-Peer Networks. In Proceedings of the 10th European Symposium on Algorithms, 2002.

[40] K. Singh and H. Schulzrinne: Peer-to-Peer Internet Telephony using SIP. In Proceeding of ACM NOSSDAV'05, pp. 63-68, June 2005.

[41] I. Beier: Design Principles of Video Conference Systems for CSCW Applications. Ph.D. thesis, Brandenburg University of Technology Cottbus, 2000 (In German).

[42] A. S. Patrick: The Human Factors of MBone Videoconferences: Recommendation for Improving Sessions and Software. JCMC 4 (1999) 3.

[43] K.Sanker, D. Futey, L. Ross and G. Brett: What is Peer-to-Peer? http://p2p.internet2.edu/documents/What%20is%20peer%20to%20peer-5.pdf.

[44] J. R. Douceur: The Sybil Attack. IPTPS'02, Mar. 2002.

[45] E. C. Popovici, R. Mahlo, M. Zuehlke, and H. Koenig: Consistency Support for a Decentralized Management in Closed Multiparty Conferences Using SIP. In Proc. of the 11th IEEE International Conference on Networks (ICON 2003), Sydney, Australia, IEEE Press, 2003, pp. 295 – 300.

[46] M. Zuehlke and H. Koenig: GCP-A Group Communication Protocol for Supporting Closed Groups in the Internet. IFIP TC 6/WG 6.7 Seventh International Conference on Intelligence in Networks (SmartNet 2002), Apr. 2002, Saariselkae, Lapland, Finland.

[47] G. V. Chockler, I. Keidar, and R. Vitenberg: Group communication specifications: A comprehensive study. ACM Computing Suryes, 4 (December 2001):427-469.

[48] CERT® Coordination Center Reports: Overview of Attack Trends http://www.cert.org/archive/pdf/attack_trends.pdf

[49] R. D. Pethia: Computer Security. CERT® Coordination Center Reports. Mar. 2000 http://www.cert.org/congressional_testimony/Pethia_testimony_Mar9.html

[50] http://www.ethereal.com/

[51] CERT® Coordination Center Reports: Overview of Internet Security. http//www.cert.org/encyc_article/tocencys.html

[52] B. Schneier: Attack trees. Dr. Dobb's Journal. Dec. 1999.

[53] F. Swiderski, and W. Snyder: Threat modeling. Microsoft Press, ISBN:0-7356-1991-3, 2004

[54] Y. Fu, J. Chase, S. Schwab, and A.Vahdat: SHARP: An Architecture for Secure Resource Peering. Proceedings of 19th ACM Symposium on Operating Systems Principles, Aug. 2003

[55] D. Chadwick: Threat Modelling for Active Directory. Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security, pp. 173-182, Sept. 2004.

[56] L. Desmet, B. Jacobs, F. Piessens, and W. Joosen: Threat Modelling for Web Services Based Web Applications. Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security, pp. 131-144, Sept. 2004.
[57] R. Shirey: Internet Security Glossary. IETF RFC 2828. May 2000.
[58] NIST: Data Encryption Standard, FIPS PUB 46, Jan. 1977.
[59] NIST: Advanced Encryption Standard, FIPS PUB 197, Nov.2001.
[60] C. Shannon: Communication Theory of Secrecy Systems. Bell Systems Technical Journal, No.4, 1949.
[61] R. Rivest, A. Shamir, and L. Adleman: A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of the ACM, Feb. 1978.
[62] W. Diffie, M. Hellman: New Directions in Cryptography. IEEE Transactions on Information Theory, Nov. 1976.
[63] J. Li: Public Key Infrastructure Technology Introduction. Intel white paper. 2000.
[64] Network Associates, Inc: How PGP works. http://www.pgpi.org/doc/pgpintro/
[65] H. Krawczyk, M. Bellare, and R. Canetti: HMAC: Keyed-Hashing for Message Authentication, RFC 2104, Feb. 1997.
[66] W. Stallings: Cryptography and Network Security, Second Edition. ISBN 0-13-869017-0, 1998 by Prentice-Hall, Inc.
[67] S. T. Kent and L. I. Millett (editors): Who Goes There? Authentication through the Lens of Privacy. CSTB Publications.
[68] W. Simpson: PPP Challenge Handshake Authentication Protocol (CHAP), IETF RFC 1994, Aug. 1996.
[69] R. M. Needham, and M. D. Schroeder: Using Encryption for Authentication in Large Networks of Computers. Communication of the ACM, 21(December 1978):993-999.
[70] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone: Handbook of Applied Cryptography, CRC Press series on discrete mathematics and its applications. CRC Press, 1997.
[71] P. Zimmermann: Pretty Good Privacy User's Guide, Volume I and II. June 1993.
[72] ITU-T Recommendation X.509 | ISO/IEC 9594-8: Public Key and Attribute Certificate Frameworks.
[73] R. Housley, W. Ford, W. Polk, and D. Solo: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459, Jan. 1999.
[74] PKI Forum: PKI Basics- a Technical Perspective. Nov. 2002 http://www.pkiforum.org/resources.html
[75] A. Abdul-Rahman: The PGP Trust Model. In EDI-Forum: The Journal of Electronic Commerce. Apr. 1997.
[76] G. Caronni: Walking the Web of Trust. Proceedings of the 9th Workshop on Enabling Technology, IEEE WET ICE'2000.

[77] G. Theodorakopolos, and J. S. Baras: Trust Evaluation in Ad-Hoc Networks. ACM Workshop on Wireless Security (WiSE'04), Oct. 2004.

[78] S. Capkun, J. P. Hubaux, and L. Buttyan: Mobility Helps Peer-to-Peer Security. EPFL-IC Technical report no. IC/2003/81.

[79] S. Capkun, L. Buttyan, and J. P. Hubaux: Self-Organized Public-Key Management for Mobile Ad Hoc Networks. IEEE Transactions on Mobile Computing, 2(2003)1.

[80] K. Berket, A. Essiari, and A. Muratas: PKI-Based Security for Peer-to-Peer Information Sharing. The 4th IEEE International Conference on Peer-to-Peer Computing, Aug. 2004.

[81] N. C. Liebau, V. Darlagiannis, A. Mauthe, and R. Steinmetz: Token-Based Accounting for P2P-Systems. 14. Fachtagung Kommunication in Verteilen Systemen(KiVS 2005), Mar. 2005.

[82] T. Grandison, M. Sloman, and I. College: A Survey of Trust in Internet Applications. IEEE Communications Surveys. Dec. 2000.

[83] ITU-T manual: Security in Telecommunications and Information Technology. Dec. 2003.

[84] Virtual Private Network Consortium (VPNC): VPN Technologies: Definitions and Requirements. VPN Consortium, Jan. 2003. http://www.vpnc.org/vpn-technologies.html.

[85] Martin Murhammer, Tim Bourne, Temas Gaidosch: A comprehensive Guide to Virtual Private Networks, Volume I: IBM Firewall, Server and Client Solutions. June 1998.

[86] MFA Forum: Detailed Analysis of Frame Relay VPNs and IP-VPNS. White paper.

[87] L. Martini, N. El-Aawar, D. Tappan, E. C. Rosen, A. Hamilton, J. Jayakumar, D. S. Vlachos, C. Liljenstolpe, G. Heron, and K. Kompella : Transport of Layer 2 Frames over MPLS. IETF, draft-martini-l2circuit-trans-mpls-16.txt, Feb. 2005.

[88] E. Rosen and Y. Rekhter: BGP/MPLS VPNs. IETF, draft-ietf-ppvpn-rfc2547bis-03.text. Oct. 2004.

[89] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little and G. Zorn: Point-to-Point Tunneling Protocol (PPTP). RFC 2637, July 1999.

[90] A. Valencia and T. Kolar: Cisco Layer Two Forwarding (Protocol) "L2F", RFC 2341. May 1998.

[91] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn and B. Palter: Layer Two Tunneling Protocol "L2TP", RFC 2661. Aug. 1999.

[92] B. Patel, B. Aboba, W. Dixon, G. Zorn and S. Booth: Securing L2TP using IPsec. RFC 3193, Nov. 2001.

[93] S. Kent and R. Atkinson: IP Authentication Header. RFC 2402. Nov. 1998.

[94] S. Kent and R. Atkinson: IP Encapsulating Security Payload. RFC 2406. Nov. 1998.

[95] N. Ferguson and B. Schneier: A Cryptographic Evaluation of IPsec. http://www.counterpane.com.

[96] D. Harkins and D. Carrel: The Internet Key Exchange (IKE). RFC 2409. Nov. 1998

[97] R. Perlman and C. Kaufman: Key Exchange in IPsec: Analysis of IKE. IEEE Internet Computing, 6(2000)4.

[98] S. Frankel, K. Kent, R. Lewkowski, A. D. Orebaugh, R. W. Ritchey and S. R. Shama: Guide to IPsec VPNs. NIST Special Publication 800-77, Jan. 2005.

[99] Nortel Networks white paper: IPsec and SSL Complementary Solutions. http://www.nortelnetworks.com/solutions/ip_vpn/collateral/nn102260-110802.pdf.

[100] M. Steiner, G. Tsudik, and M. Waidner: CLIQUES: A New Approach to Group Key Agreement. IEEE ICDCS, 1998, pp. 380-397.

[101] Y. Kim, A. Perrig, and G. Tsudik: Simple and Fault-tolerant Key Agreement for Dynamic Collaborative groups. ACM CCS 2000, pp. 235–244.

[102] O. Rodeh, K. P. Birman, D. Dolev: Optimized Group Rekey for Group Communication Systems. NDSS 2000, pp. 39-48.

[103] F. Liu and H. Koenig: Secure and Efficient Key Distribution for Collaborate Applications. IEEE Collaboratecom 2005, Dec. 2005.

[104] Chung-Ping Wu and C.-C. Jay Kuo: Speech Content Authentication Integrated with CELP Speech Coders. IEEE ICME 2001, pp. 1009-1012, Aug. 2001.

[105] Chung-Ping Wu and C.-C. Jay Kuo: Comparison of Two Speech Content Authentication Approaches. In Proceedings of SPIE Volume 4675. Security and Watermarking of Multimedia Contents IV, pp. 158-169, Jan. 2002.

[106] M. Steinebach and J. Dittmann: Watermarking-Based Digital Audio Data Authentication. EURASIP Journal on Applied Signal Processing 2003:10, pp. 1001-1015.

[107] Q. Sun and S. Ye: A Crypto Signature Scheme for Image Authentication over Wireless Channel. International Journal of Image and Graphics, 5(2005)1:1-14.

[108] B. G. Haskell, A. Puri, and A. N. Netravali: Digital Video: An Introduction to MPEG-2. Kluwer Academic Publishers, 1996.

[109] F. Liu and H. Koenig: A Novel Encryption Algorithm for High Resolution Video. In Proceeding of ACM NOSSDAV'05, pp. 69-74, June 2005.

[110] F. Liu and H. Koenig: Puzzle-A Novel Video Encryption Algorithm. IFIP CMS 2005, Springer LNCS 3677, pp. 88-97, Sept. 2005.

[111] A. S. Patrick: The Human Factors of MBone Videoconferences: Recommendation for Improving Sessions and Software. JCMC 4 (1999) 3.

[112] J. Snoeyink, S. Suir, and G. Vorghese: A Lower Bound for Multicast Key Distribution. IEEE INFOCOM 2001, pp. 422-431.

[113] P. McDaniel, A. Prakash, and P. Honeyman: Antigone: A Flexible Framework for Secure Group Communication, CITI Technical Report 99-2, University of Michigan, Sept. 1999.

[114] P. S. Kruus: A Survey of Multicast Security Issues and Architectures. 21st National Information Systems Security Conference (NISSC), Oct. 1998. http://csrc.nist.gov/nissc/1998/proceedings/paperF10.pdf.

[115] S. Rafaeli and D. Hutchison: A Survey of Key Management for Secure Group Communication. ACM Computing Surveys 35(2003)3:309-329.

[116] T. Ballardie, P. Francis, and J. Crowcroft: Core based trees: An Architecture for Scalable Interdomain Multicast Routing. Proceedings of ACM SIGCOMM'93(1993), pp. 85-95.

[117] H. Harney and C. Muckenhirn: Group Key Management Protocol (GKMP) Specification, July 1997, RFC 2094.

[118] M. Burmester and Y. Desmedt: A Secure and Efficient Conference Key Distribution System. In Advances in Cryptology EUROCRYPT'94, Springer LNCS 950, 1995, pp. 275-286.

[119] M. Steiner, G. Tsudik, and M. Waidner: Key Agreement in Dynamic Peer Groups. IEEE Transactions on Parallel and Distributed Systems (August 2000).

[120] C. Wong, M. Gouda, and S. Lam: Secure Group Communication Using Key Graphs. IEEE/ACM Transactions on Networking 8(2000)1:16-30.

[121] D. A. McGrew and A. T. Sherman: Key Establishment in Large Dynamic Groups using One-way Function Trees. Tech. Rep. No.0755 (May), TIS Labs at Network Associates, Inc., Glenwood, Md.

[122] M. J. Moyer, J. R. Rao and P. Rohatgi: Maintaining Balanced Key Trees for Secure Multicast. Technical report, IETF, June 1999. draft-irtf-smug-key-tree-balance-00.txt.

[123] Y. Kim, A. Perrig, and G. Tsudik: Tree-based Group Key Agreement. ACM Transactions on Information Systems Security (TISSEC) 7(2004) 1: 60-96.

[124] E. Rescorla: Diffie-Hellman Key Agreement Method. RFC 2631, June 1999.

[125] L. Dondeti, S. Mukherjee, and A. Samal: Disec: A Distributed Framework for Scalable Secure Many-to-Many Communication. In: Proceedings of the Fifth IEEE Symposium on Computers and Communications (ISCC 2000), July 2000.

[126] G. Ateniese, M. Steiner, and G. Tsudik: New Multiparty Authentication Services and Key Agreement Protocols. IEEE Journal Selected Areas in Communication 18 (2000) 4.

[127] O. Pereia and J. J. Quisquater: A Security Analysis of Cliques Protocol Suites. In: Proceedings of the 16th International Conference on Information Security: Trusted information: the new decade challenge. Pairs, France. 2001, pp. 151-166.

[128] S. Mittra: Iolus: A Framework for Scalable Secure Multicasting. in Proc. ACM SIGCOMM, Cannes, France, Sept. 1997, pp. 277–288

[129] B. Whetten, T. Montgomery, and S. Kaplan: A High Performance Totally Ordered Multicast Protocol. International Workshop on Theory and Practice in Distributed Systems, Springer LNCS 938, pp. 33-57, 1994.

[130] D. A. Agarwal: Totem: A Reliable Ordered Delivery Protocol for Interconnected Local Area Networks. Ph.D. Thesis, University of Santa Barbara, Dec. 1994.

[131] K. Birman, R. Constable, M. Hayden, C. Kreitz, O. Rodeh, R. van Renesse, W. Vogels: The Horus and Ensemble Projects: Accomplishments and Limitations. Proc. of the DARPA Information Survivability Conference & Exposition (DISCEX '00), Hilton Head, South Carolina, 2000.

[132] Y. Amir, C. Danilov, and J. Stanton: A Low Latency, Loss Tolerant Architecture and Protocol for Wide Area Group Communication. In Proc. 30th IEEE FTCS, June 2000.

[133] F. Liu and H. Koenig: A Token Based Key Distribution Protocol for Closed Group Meetings. Proceeding IFIP Netcon 2005, Nov. 2005

[134] C. Kaufman, Internet Key Exchange (IKEv2) Protocol, draft-ietf-ipsec-ikev2-17.txt, Sept. 2004.

[135] R. Canetti and H. Krawczyk: Security Analysis of IKE's Signature-based Key Exchange Protocol. Crypto 2002. Springer LNCS 2442, pp. 143-161, 2002.

[136] H. Krawczyk: SIGMA: The 'SIGn-and-MAC' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols. Crypto 2003. Springer LNCS 2729, pp. 400-425, 2003.

[137] A.Krywaniuk: Security Properties of IPsec Protocol Suite. Draft-ietf-ipsec-properties-02-txt. June 2002

[138] OpenSSL project. http://www.openssl.org/

[139] M. Baugher, D. McGrew, M. Naslund, E. Carrara and K. Norrman: The Secure Real-time Transport Protocol (SRTP). RFC 3711, Mar. 2004.

[140] D. L. Mills: Network Time Protocol (Version 3) Specification, Implementation and analysis. RFC 1305, Mar. 1992

[141] Digital Equipment Corporation: Digital Time Service Functional Specification Version T.1.0.5, 1989

[142] R. Krauz: Integration of a Key Management into the Video Conference System BRAVIS. Diplom thesis, Brandenburg University of Technology Cottbus, 2005 (In German).

[143] M. Zuehlke, and H. Koenig: A Signaling Protocol for Small Closed Dynamic Multi-peer Groups. In Z. Mammeri and P. Lorenz (eds.): High Speed Networks and Multimedia Communications (HSNMC 2004), Springer LNCS 3079, pp. 973-984, 2004.

[144] ISO/IEC 11172-2: Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5 Mbit/s: Video. 1993.

[145] ISO/IEC 13818-2: Information Technology--Generic Coding of Moving Pictures and Associated Audio Information: Video. 2000.

[146] ITU-T Recommendation H.261: Video Code for Audiovisual Services at px64 kbit/s. Mar. 1993.

[147] ITU-T Recommendation H.263: Video Coding for Low Bit Rate Communication. Feb. 98.

[148] W. Effelsberg and R. Steinmetz: Video Compression Techniques. Dpunkt-Verlag, 1998.

[149] I. Agi and L. Gong: An Empirical Study of MPEG Video Transmission. In Proceedings of the internet Society Symposium on Network and Distributed System Security, pp. 137-144, 1996.

[150] G. Cote and L. Winer: Recent Advances in Video Compression standards. IEEE Canadian Review, 2002.

[151] ISO/IEC 10918-1: Information technology--Digital Compression and Coding of Continuous-tone Still Images: Requirements and guidelines. 1994.

[152] D. A. Huffman: A Method for The Construction of Minimum Redundancy Codes, Proceedings of IRE 40(9):1098-1101, 1952.

[153] I. H. Witten, R. M. Neal, and J. G. Cleary: Arithmetic Coding for Data Compression, Communications of the ACM, pp. 520-540, June 1987.

[154] P. D. Symes: Video Compression. McGraw-Hill Companies, 1998.

[155] B. Fuhrt and D. Kirovski: Eds. Multimedia Security Handbook, CRC Press, 2004.

[156] X. Liu and A. M. Eskicioglu: Selective Encryption of Multimedia Content in Distribution Networks: Challenges and New Directions. IASTED International Conference on Communications, Internet and Information Technology (CIIT), 2003.

[157] W. Zeng and S. Lei: Efficient Frequency Domain Selective Scrambling of Digital Video. IEEE Transactions on Multimedia, 2002.

[158] L. Tang: Methods for Encrypting and Decrypting MPEG Video Data Efficiently, ACM International Conference on Multimedia, pp. 219-229, 1996.

[159] L. Qiao and K. Nahrstedt: Is MPEG Encryption by Using Random List Instead of Zigang Order Secure? IEEE International Symposium on Consumer Electronics, pp. 226-229, 1997.

[160] C. Shi, S. Y. Wang and B. Bhargava: MPEG Video Encryption in Real-Time Using Secret Key Cryptography. 1999 International Conference on Parallel and Distributed Processing Techniques and Applications (PDATA'99), June 1999.

[161] C. -P. Wu and C.-C. J. Kuo: Fast Encryption Methods for Audiovisual Data Confidentiality. SPIE International Symposia on Information Technologies 2000, 4209 (2000): 284-295.

[162] C. -P. Wu and C.-C. J. Kuo: Efficient Multimedia Encryption via Entropy Codec Design. SPIE Int. Symposium on Electronic Imaging 2001, Vol.4314, 2001.

[163] S. Li, G. Chen, A. Cheung and K. T. Lo: Cryptanalysis of an MPEG-Video Encryption Scheme Based on Secret Huffman Tables.

http://arxiv.org/abs/cs.MM/0509035.
[164] G. A. Spanos and T. B. Maples: Performance Study of a Selective Encryption Scheme for the Security of Networked Real-time Video. ICCCN, pp. 2-10, 1995.
[165] Y. Li, Z. Chen, S. M. Tan, and R. H. Campbell: Security Enhanced MPEG Player. IEEE 1st International Workshop on Multimedia Software, 1996.
[166] L.Qiao and K. Nahrstedt: Comparison of MPEG Encryption Algorithms. Computer and Graphics 22 (1998) 4:437-448
[167] P. Rogaway and D. Coppersmith: A software-optimized encryption algorithm. Proceedings of the 1st International Workshop on Fast Software Encryption, Springer, pp. 56-63, 1993.
[168] M. Dworkin: Recommendation for Block Cipher Modes of Operation, Methods, and Tech-niques. NIST Special Publication 800-38A, Dec. 2001.
[169] RSA Laboratories: Frequently Asked Question about Today's cryptography, version 4.1.
[170] Demo MPEG-1 clip: "Table tennis". http://bmrc.berkeley.edu/frame/research/mpeg/media/ts.mpg.
[171] MPlayer: http://www.mplayerhq.hu.
[172] Sun Microsystems, Inc: SunVideo Plus 1.3 User's Guide. 1999.
[173] R. Schneier: Applied cryptography. New York. Wiley, 1996.
[174] E. Biham and A. Shamir: Differential Cryptanalysis of the Full 16-round DES. Advances in Cryptology-CRYPTO'92. Springer, pp. 487-496, 1992.
[175] M. Stanek and L. Stanekova: Unpuzzling Puzzle (analysis of a video encryption algorithm). IEEE the Second International Workshop on Security in Networks and Distributed Systems (SNDS-06), Apr. 2006.
[176] http://en.wikipedia.org/wiki/Distributed.net
[177] A. Rowstron and P. Druschel: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In Proceedings of IFIP/ACM Middleware, 2001.
[178] http://en.wikipedia.org/wiki/Videoconference
[179] D. Bryan, B. Lowekamp, and C. Jennings: A P2P Approach to SIP Registration and Resource Location. IETF draft-bryan-sipping-p2p-02.txt. Mar. 2006.
[180] The ViDe Videoconferencing Cookbook. http://www.videnet.gatech.edu/cookbook.en/
[181] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman: Role-based Access Control Models. IEEE Computer 29, 2(Feb.), 38-47.
[182] F. Liu, and H. Koenig: A Secure P2P Video Conference System for Enterprise Environments. IFIP NPC 2005, Springer LNCS 3779, pp. 88-96, Dec. 2005.