Key words: Data Security, Security Architecture, Data Communication, Open Systems, OSI, FTAM

Abstract

An architecture for building secure open systems communicating via untrusted global data networks is presented. The work is based on the ISO OSI reference model and its security architecture addendum but the results are directly applicable to virtually any layered communications architecture such as the DoD Internet architecture (TCP/IP). Also, several extensions and modifications to the OSI model are suggested.

A semiformal model for the security of communications between peer entities within an OSI layer and between entire application instances is presented. This model, along with some new concepts introduced, is successfully used to analyze the security functions of open systems. This work brings a whole new research area at the boundary between formal protocol research and cryptology within the reach of formal study. Complete formalization of and further elaboration on the model are beyond the scope of this work and left for further study.

The degree of freedom offered by the OSI Security Architecture in placing the security services and mechanisms defined in the OSI reference model into its layers is restricted in such a way as to not affect the security of the over-all communication service and well-founded placements for various services and mechanisms are proposed.

In order to demonstrate the implementability of the solutions proposed here, actual low-level security mechanisms are chosen to fill in the missing parts. Finally, a communications profile for secure FTAM is drawn up and a subset of it is implemented based on existing OSI implementations.

Foreword

This paper is a cosmetically updated version of my Doctoral Dissertation completed in late 1990 and defended in March 1991 at Helsinki University of Technology.

This work is based on the experience of several years of protocol implementation work with the Technical Research Centre of Finland and my more recent experience in the development of value added services for our customers at Telecom Finland. The approach is rather pragmatic (for a dissertation) and the problems are taken from real life. I hope this paper will serve a purpose in raising the general level of consciousness of the security risks of open systems and stimulating discussion and research in the area of developing secure open systems.

I hope my possible American readers will not be scared off by the seeming OSI-orientation of this work. I am by no means an OSI pedantic and most of the results obtained here are directly applicable to other layered communications architectures such as the DoD Internet (TCP/IP) architecture.

I wish to thank my reviewers Dr. Kaisa Nyberg of the Finnish Army and Dr. Olli Martikainen of the Technical Research Centre of Finland for their constructive criticism and many comments on the draft versions of this paper. I also thank professor Jarmo Harju of Lappeenranta University of Technology for the many discussions on this topic at the early stage of this work. I thank my opponent professor David Farber of University of Pennsylvania, who urged me make these minor changes to my work and make it publicly available through the internet, for his careful examination of my thesis and the stimulating conversations with him in March 1991.

Comments, suggestions, corrections and critique is welcome.

Helsinki, Finland, May 31, 1991

Arto T. Karila

Contents

1 Introduction 1

1.1 Evolution towards Open Networks 1

1.2 On the Security of LANs 5

1.3 Need for Security in Open Systems 6

1.4 Standardization of Open Systems Security 7

1.5 The Motivation and Goal of This Work 9

2 The Basis and Scope of This Work 12

2.1 The Basic Model 12
2.1.1 End Systems and Application Instances 13
2.1.2 Network 15
2.1.3 Trusted Parties 15
2.1.4 The Outside World 16

2.2 Communications Categories 16

2.3 The Scope of This Study 19

2.4 Approach to the Goals 21

3 On Security Requirements and Functions 22

3.1 A Security Model 23

3.2 Security Requirements 27
3.2.1 Security Requirements of FTAM 28
3.2.2 Security Requirements of the Directory 29
3.2.3 Security Requirements of the Message Handling System 31
3.2.4 Summary of the Security Requirements 33

3.3 Security Functions 33

3.4 On the Formal Analysis of Secure Open Systems 38

3.5 On the Placement of Security Functions 40
3.5.1 Application Process 42
3.5.2 Application and Presentation Layers 42
3.5.3 The Lower Layers 47

4 A Secure General Purpose OSI Protocol Stack 49

4.1 Authentication 51
4.1.1 Authentication and its Significance with Various Services 51
4.1.2 Placement of Authentication in the OSI model 52
4.1.3 Authentication Mechanisms 52

4.2 Integrity 53
4.2.1 Integrity and its Significance with Various Services 53
4.2.2 Placement of Integrity in the OSI Model 53
4.2.3 Integrity Mechanisms 54

4.3 Confidentiality 56
4.3.1 Confidentiality and its Significance with Various Services 56
4.3.2 Placement of Confidentiality in the OSI Model 56
4.3.3 Confidentiality Mechanisms 57

4.4 Non-Repudiation 57
4.4.1 Non-Repudiation and its Significance with Various Services 57
4.4.2 Placement of Non-Repudiation in the OSI Model 58
4.4.3 Non-Repudiation Mechanisms 58

4.5 Access Control 59
4.5.1 Access Control and its Significance with Various Services 59
4.5.2 Placement of Access Control in the OSI Model 60
4.5.3 Access Control Mechanisms 60

5 Management Issues 61

5.1 Activation and Negotiation of Security Functions 61

5.2 Distribution and Management of Keys 62
5.2.1 Management of Master Keys 63
5.2.2 Management of Session Keys 65
5.2.3 On the Generation of Public Key Pairs 66

6 Implementational Issues 67

6.1 Software Environments 67
6.1.1 Some Sample Environments 67
6.1.2 The Effects of Software Environment on Systems Security 70

6.2 On the Use of Hardware and Software 71

6.3 On the Role of Smart Cards 71

6.4 Real Cryptosystems 75

7 Implementing Secure FTAM 76

7.1 Requirements and Restrictions 76

7.2 FTAM Security Profile 76

7.3 Management Issues 78

7.4 Implementation Environment 79

7.5 Current Status of the Project 79

8 Conclusions 82

Appendix:

A1 Security Mechanisms 84

A1.1 Data Encryption Mechanisms 85
A1.1.1 Symmetric Encryption Mechanisms 86
A1.1.2 Asymmetric Encryption Mechanisms 87
A1.1.3 On the Use of Encryption Mechanisms 87

A1.2 Signature Mechanisms 88

A1.3 Integrity Mechanisms 89

A1.4 Authentication 93

A1.5 Zero-Knowledge Techniques 102

A1.6 Physical Mechanisms 103

A2 Some Commonly Used Cryptosystems 105

A2.1 DES 105

A2.2 RSA 107

A2.3 The Diffie-Hellman Algorithm 108

A3 The Kerberos Authentication Server 109

A4 Security Policy 111

Bibliography 113

Glossary 132

1 Introduction

The role of information networks in today's world is becoming ever more central. Information systems are taking over new areas of application and replacing manual systems in private, public and corporate life. Electronic Data Processing (EDP) is no longer seen as simply a way to rationalize the internal operations of an organization but it is expanding from within the organization to its external relations, such as those with customers, partners, suppliers and officials.

Data communication is an integral part of any modern information system. In the earlier days data communication was seen as a way of making computing equipment and centralized information systems accessible from a geographically large area. Nowadays the network is seen as the central part of information systems, interconnecting communicating peers, not masters and slaves.

The client-server architecture is based on the central role of telecommunications networks. The network interconnects users with various information services (and these services with one-another). Large, distributed information systems are increasingly often collections of smaller systems communicating through the network (not necessarily in real-time).

1.1 Evolution towards Open Networks

Separate networks are integrating into a universal internet, consisting of a number of interconnected networks. The *International Organization of Standardization* (*ISO*) *Connectionless Network Protocol* (*CLNP*, also called *ISO IP*) will define a way to interconnect virtually all the networks and access any *Network Service Access Point* (*NSAP*) from any other NSAP worldwide (see e.g. [BI89, IS8348/A1, IS8473] for detailed information on ISO IP).

At the moment, the *Internet Protocol Suite* (also known as *TCP/IP*, see [Cla88] for an overview of the goals and reasoning behind the design of these protocols) of the U.S. Department of Defense (DoD), defined in the *Request for Comment* (*RFC*) standards, offers the most natural evolution path towards true open systems (in the OSI sense of the word). The DoD IP network protocol [RFC81a] offers the first usable way of interconnecting heterogeneous networks and routing traffic between them. It is the *Connectionless Network Service* (*CLNS*) [IS8348/A1] and *Connectionless Network Protocol* (*CLNP*) [IS8473] of ISO that are the natural successors of DoD IP, offering practically the same functionality but in an internationally standardized form and with a larger, better structured address space.

The employment of US GOSIP [GOS89] in August 1990, mandating the use of ISO IP by US Agencies, whenever possible, will give significant boost to ISO IP during the year 1991. The proliferation of DECnet Phase V, based on ISO IP and TP4 [IS8073], coincides with this development. [Hei90] gives a concise overview of the migration from DoD IP to ISO IP.

In addition to the Connectionless Network Service and Protocol, ISO also defines the Connection-mode Network Service (CONS) [IS8348, X.213] and its associated protocols. Network or transport layer gateways can be used to interface between networks offering these two kinds of service.

It is, however, probable that the connectionless version of the ISO Network Service will be predominant in the near future, e.g. for the following reasons:

- For the client-server architecture connectionless communications are more natural than connection-oriented communications. In many cases, such as with directory or database queries, the exchange of one request-response pair of messages is sufficient to do the job without the unnecessary overhead of opening and closing a (virtual) network connection.

- For example LANs, which are becoming ever more popular as local network solutions, are connectionless by nature. Mapping a connectionless service onto a connection-oriented network is much easier than doing the opposite.

- It has been shown in the world-wide Academic Internet (currently based on the DoD IP) that it is possible to build large, technologically heterogeneous networks offering universal connectionless service while using low-cost routers and achieving high line utilization. In fact, connectionless service appears to make the interconnection of networks of highly different capacities rather straight-forward and efficient.

- Transmission techniques are becoming ever more reliable making it feasible to recover from transmission errors at a higher protocol layer, namely by using the ISO *Class 4 Transport Protocol* (*ISO TP4*) [IS8073, X.224] or the DoD *Transmission Control Protocol* (*TCP*) [RFC81b], which effectively hide the connectionless nature of the network service from the upper protocol layers.

In a world-wide connectionless internet alternate routes, load balancing and automatic rerouting are operating on a global scale. Individually routed data packets can be expected to arrive via any route and it is impossible to trust all the networks along the way. Therefore, it is practically impossible to guarantee the security of the network, even though teleoperators will do their best to make their networks reasonably secure.

For example [Bar90] deals with the security requirements of WANs. This paper only discusses the more traditional networks: the Public Switched Telephone Network (PSTN), its digital successor Integrated Services Digital Network (ISDN), and the X.25 packet network [X.25]. The paper is interesting because it gives a list of basic requirements for the security of a WAN as seen from the teleoperator's perspective. A rather superficial survey of the security problems associated with a DoD IP network is presented in [Bel89] (and criticized in [Ken89]). In practice, a world-wide internet will always have to be assumed untrustworthy.

The DoD TCP/IP offers to a great extent the same kind of service as ISO TP4/CLNP. Upper layer ISO protocols can be run on top of either, as demonstrated by the example of ISODE, and the evolution from DoD to ISO protocols will be aided by transport layer gateways [KLP83, Ros90, LT90] (for an example of an asymmetric transport layer gateway see figure 7-3). Some of the first large scale open systems will undoubtedly be based on the existing DoD IP infrastructure. During the migration towards OSI the upper layer protocols of the DoD Internet Protocol Suite will also be run on top of OSI lower layers, where ISO IP solves the problems of insufficient and badly structured address space of the DoD IP. As OSI matures, true seven-layer open systems (in the ISO sense of the word) will emerge.

In the future, Corporate Networks, Value Added Networks (VANs) and such will not be separate physical networks but rather virtual networks, i.e. collections of Network Service Access Points (NSAPs) forming a logical network. Also, one NSAP can simultaneously belong to any number of such logical networks.

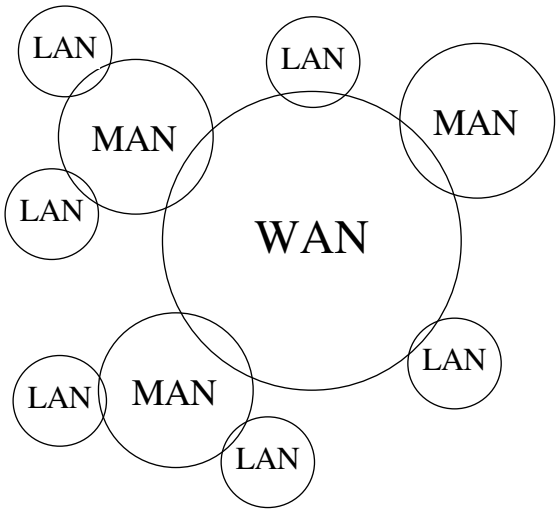

Figure 1-1: A multi-level network hierarchy.

A multi-level hierarchy of networks is emerging (as depicted in figure 1-1):

Individual computers and work stations are connected to fast Local Area Networks (LANs) spanning office buildings or parts of them. LANs offer transmission speeds in the range of Mb/s today and the speeds are rising. This is a significant improvement over the more traditional transmission speeds in the order of kb/s. LANs also offer full connectivity (direct communication between any two stations in the same network). Examples of todays LANs are Ethernet (10 Mb/s) [IS8802-3], Token Ring (4 or 16 Mb/s) [IS8802-5], and FDDI and FDDI II (100 Mb/s) [IS9314-1,2,3,4,5, RHF90].

These LANs may be connected into fast areal backbone networks interconnecting LANs within e.g. one building, building complex or campus area at a speed comparable to that of the LANs. Current backbones are of the same types as the LANs but usually built with fiber optics technology (see e.g. [Kat90]).

Metropolitan Area Networks (MANs) [IEE90] are emerging. They will span entire cities with speeds above 100 Mb/s. LANs and Private Branch Exchanges (PBXs) can in the near future be connected to these MANs. Unlike LANs, MANs will be an integral part of the modern public network infrastructure being owned and operated by teleoperators and sharing the addressing and network management schemes of other public networks.

Wide Area Networks (WANs) are used for long-haul services. Modern WANs are based on fiber optics transmission systems in the Gb/s speed range. 2 Mb/s leased circuits are now available to most places, even internationally. Among the most interesting new public data services are *Frame Relay*, a next generation public packet service standardized by CCITT and ANSI, already commercially supported and soon emerging in many countries, and the *Switched Multi-megabit Data Service* (*SMDS*), a fast packet service expected to supersede the current WAN services [DKK90, PK90]. The long-haul network is (possibly) evolving towards Asynchronous Transfer Mode (ATM) and Broadband ISDN. For those interested in the evolution of wide-area networks [Hei91] gives a good overview of the current trends in this area.

In this network hierarchy most of the traffic is local (within the same LAN). The LAN offers virtually unlimited capacity for a flat fee (the cost of the investment and operation). As we rise higher in the hierarchy the amount of traffic decreases and the cost of transmission increases but the quality and speed of the service remain high. A few orders of magnitude increase in the capacities of internets is expected within a couple of years, as pointed out e.g. in [Par90].

This development opens completely new views for designers of information services. For example, a work station connected to a LAN can communicate with virtually any

other computer (world-wide) at a high bandwidth and with a relatively short round-trip delay. It is clear that this kind of connectivity brings with it hitherto unthought-of security risks that need to be taken under control.

The complexities of network management are described e.g. in [CPW89]. The emerging new network architecture presents two major challenges for telecommunications operators and vendors of information systems:

- Network management needs to be raised onto a new level, where virtual networks can efficiently be defined and managed on top of the integrating basic networks.

- Tools for enforcing the security policies dictated by owners of these virtual networks are needed.

Practically all Local Area Networks have a multicast capability where the same (data link level) message can be simultaneously addressed to a group of recipients. This capability is also included in some of the modern WAN networks techniques, such as Frame Relay. Also some applications, such as MHS, include a multicast capability. Unfortunately the ISO network service is not designed to offer this service. For emerging new internet-based services, such as workstation-based multimedia- and video conferencing, a multicast network service would be most useful.

In fact, there is a strong tendency to develop a multicast capability to protocols of various layers of the OSI model. The need for LAN-like multicasting in internets is pointed out e.g. in [Dee88] and various data link and network layer multicast routing algorithms are suggested. A multicast transport protocol, supporting a sequence of exchanges of arbitrarily sized request and response messages between a client a large number of servers, is proposed in [CP88].

1.2 On the Security of LANs

While LANs have many desirable properties, such as high bandwidth, short transmission delay, and no traffic-dependent cost, they also pose a considerable security threat.

In a LAN, all the stations are constantly listening to the medium, picking up all transmitted messages, and recognizing those addressed to them by the destination addresses of the messages. This means that any station can undetected listen to all the traffic in a LAN. Also, any station can assume a false identity and start transmitting with a wrong source address thereby impersonating as another station.

One of the most obvious attacks is the recording and replay of authentication sequences. With a workstation costing about $1,000 (in 1990) and widely available public domain software, anybody with access to the LAN (such as Ethernet or Token Ring) can monitor the traffic, search for sequences such as "*Username:*" and "*Password:*" on virtual terminal connections, record the responses to these prompts, and later log in a service under a false identity.

It is easy to imagine more ingenious attacks through a LAN. To protect against these threats the IEEE 802.10 work group is standardizing the use of data link level encryption in LANs but this work is still at an early stage. In spite of what has been said here about the security of LANs, it must be noted that any large network offering good connectivity between organizations and countries has to be considered untrusted and, on the other hand, a piece of LAN along the route from A to B can be made reasonably secure by using a pair of trusted encryption devices.

The question remains, how to protect against the threats coming from the network. Should we abandon the possibilities offered by new networking technologies securing information systems by isolating them? Obviously not! The most logical solution is to reject the idea of networks being secure and build systems in such a way that their security does not rely on the security of the network they are built on. Today's cryptographic techniques provide us with the tools for building such systems. What is needed is significant architectural work in defining and building secure seven-layer open systems from the more basic building blocks available today.

1.3 Need for Security in Open Systems

Connectivity and security are inherently contradictory requirements. However, openness does not mean lack of security but interconnectivity and the ability to interoperate between systems in different organizations and from different manufacturers. It is possible to build open systems with any required level of security. When building a distributed information system, it is essential that we be able to define precisely what users and end systems are allowed what kind of access to each service and be able to enforce this policy in practice. It is also often necessary that we be able to make commitments electronically (and confidentially) and be able to show them true afterwards.

Until recently, international standardization has focused on various aspects of interconnecting information systems and making them interoperate with a sufficient level of functionality. One of the major obstacles hindering the proliferation of EDI, and other types of inter-corporate utilization of Electronic Data Processing (EDP), is the lack of security in today's distributed information systems. The basic set-up is that of parties with at least some degree of confidence in one-another wishing to securely communicate through an untrustworthy network. The need for security in open systems has

recently been officially acknowledged on many forums, e.g. in [Com90]. Another important recent document on the security of EDI is [TED90b].

1.4 Standardization of Open Systems Security

The ISO *Open Systems Interconnection* (*OSI*) reference model [IS7498-1, IS7498/C1, X.200] has recently become universally accepted as the basis of open systems. All the major manufacturers of computer systems are committed to OSI and some of the most important users, such as the DoD, have based their data communications strategies on it. Despite its serious shortcomings, which will be discussed later in this study, the OSI reference model is a lot better than no commonly agreed on reference model at all and we shall be forced to live with it.

From the beginning OSI has concentrated on interoperability leaving management and security issues with little attention. The OSI model was later extended with connectionless-mode communications, security, naming and addressing, and management, which were added to the OSI model as its later parts and addenda [IS7498-2,3,4, IS7498/A1,A2]. Unfortunately these extensions are still not nearly as mature as the basic OSI reference model.

The OSI Security Architecture [IS7498-2] addresses security by listing a number of security threats and defining a number of *Security Services* (or more properly *Security Functions*, see 3.1) and mechanisms to protect against these threats. The threat model assumed in this work is that of [IS7498-2]. The addendum defines possible places for these services and mechanisms in the seven layers of OSI but still leaves a lot of open ends. Similar work has been carried out in ECMA [ECM88, ECM89].

Among the shortcomings of the OSI Security Architecture are the following:

- Most services can be placed at any one of a number of layers (especially at layers 3, 4, 6 and 7). This freedom needs to be restricted in a way that will not affect the overall security of the system.

- Little, if any, clue is given as to how these services and mechanisms should interoperate with the rest of the system. E.g. how they are embedded into existing services and protocols, how their use is negotiated, and how they are activated.

- Security-wise, the end system needs to be considered as a whole. From the user's point of view the security of the service provided by the entire communications system, not that provided by each layer, is essential. In this respect, the

layers are strongly interdependent. Clearly, an integrated view on open systems security is wanted.

An overview of the standardization of open systems security is given in [CEN90].

The OSI model defines only a framework for the standardization of open systems. Numerous other OSI standards standards define services and protocols for various layers of OSI. Standardization of actual Open Systems can be seen as consisting of two phases.

First, the need for standardizing some functional entity, such as message handling, is realized. This standardization leads into a number of standards, in the case of message handling the X.400-series recommendations of the CCITT. However, these *base standards* often include parallel, alternative or optional features included in the standard as a compromise between various parties. In various countries and by various manufacturers different functional groups and subsets of these standards are adopted. Therefore, compliance with the standards does not alone guarantee interoperability between two end systems. This was first seen in practice with the public packet networks of the type X.25, where interfaces to the network still vary from country to country and teleoperator to teleoperator, even though these networks nowadays are globally interconnected.

In the second phase, *functional standards* (or *profiles*), referring to the base standards and specifying in detail the functional groups, functions, options, PDU sizes etc. used, are defined. The purpose of OSI profiles is to define how various OSI base standards are to be used in specific, real contexts to implement systems capable of interoperating. When true interoperability between systems by various vendors (which is the ultimate goal of the whole OSI philosophy) is required, it is usually better to refer to profiles and require that the systems comply with them than refer to the base standards. For example, if a message handling system is guaranteed to comply with the European MHS profiles (CEPT A/311 and CEN/CENELEC A/3211) it either will interoperate with other similar products or one of the vendors can be pointed to be at fault.

In OSI standardization defining communications profiles for various applications has recently become perhaps the most important area of activity. A new type of document called the *International Standardized Profile* has been created (see [IS10000-1,2]). However, it is only after *Security Profiles*, defining in detail the security functions and mechanisms of complete seven-layer protocol stacks and their use, have been defined that truly open secure systems can be implemented.

I claim that defining Security Profiles will prove to be a major step on the way towards secure open systems. The suggested profile for FTAM (depicted in figure 7-1) as well as the results presented in chapter 4 are among the first attempts to this direction.

1.5 The Motivation and Goal of This Work

It is seen that definitions and actual implementations of secure open systems are vital for the future of information networks. Extensive research has been done (and is being done) in the area of cryptology resulting in usable cryptographic mechanisms, such as symmetric and asymmetric cryptosystems, hash functions, and signature and authentication schemes. Also, the OSI reference model [IS7498-1] has been extended to include a security framework [IS7498-2]. The increasing activity in ISO in security architectures is demonstrated by a number of new standards in this area, such as [ISO90c] and [IS10181-1].

However, little work has still been done between these two extremes on the abstractional scale. The OSI reference model is on a very high level of abstraction, offering us a conceptual framework but little guidance for building real systems. On the other hand, most of the work done in the field of cryptology concentrates on the mathematical properties of individual cryptosystems. At most, authentication protocols and such have been analyzed with no regard to their position in the OSI model and relation to the rest of the open system. One of the first mentions of these two, often disjoint, tracks of development is found in [MM83]. The final report of the COST-11 Ter project [COS90] (chaired by S. Muftic), which is soon coming out, is among the first publications in the area of architectures for secure open systems.

There is an urgent need for work combining these two main lines of research in the area of secure communications. Starting from the general models and frameworks, security functions, services and mechanisms need to be put in their proper places in open systems. They need to be embedded into the existing protocols and services, interfaced with management etc. Finally, appropriate cryptosystems need to be adopted to fill in the missing parts. As a result of this work, justified, implementable solutions to the security problems of open systems now exist.

This work combines the current knowledge of open systems, the OSI framework, open systems management, actual communications protocols, the available cryptographic functions, and their implementation, coming up with an architecture for building secure open systems. Because this work aims at actual working implementations of secure open systems, implementational issues will be kept in mind throughout the study.

This study is written for the OSI-oriented reader involved in research and development of secure open systems. The OSI reference model is used as a guideline, even though a critical view is kept on it. Based on the reasoning given in chapter 3, it can be claimed that security should be an integral part of the service provided by the OSI world and embedded into the layers of OSI, rather than something designed and implemented

from scratch for each application using the services of OSI. This work is not very closely tied to the OSI model which is merely used as a frame of reference. In fact the results obtained here can be directly applied to virtually any layered communications architecture, such as the DoD Internet Protocol Suite or IBM's System Networks Architecture (SNA).

The mathematical properties of protocols as well as mathematical cryptology have been areas of theoretical research for a long time. There are certain well known requirements that can be posed to any telecommunications protocol, such as: completeness, fairness, and freedom of dead-locks. Also cryptology has been an area of formal theoretical study ever since Shannon published his now classical paper breaking new ground in 1949 [Sha49]. Some of these aspects are discussed in some classical papers (such as [NS78], [DS81] and [NS89]). However, so far a list of the properties of "good" secure protocols have not been defined.

The main goals of this work are to create an architectural framework integrating the security functions and mechanisms into the the OSI reference model and to analyze various security functions and mechanisms with respect to this framework. Several new concepts, such as *security context* and *security profile*, are introduced and used successfully in analyzing the security requirements of open systems and designing solutions for them.

This work gives the guidelines for an entire new research area at the boundary between theoretical protocol research and cryptology, which so far has been virgin soil. The importance of this area is demonstrated by the fact, that in this study alone a number of new research topics are pointed out and a number of problems, until now completely unrecognized, are formulated and brought within the reach of theoretical study.

In chapter 2, the scope of this work and the model of the world this work is based on are defined.

In chapter 3, a formal security model is defined, which is then used as a guideline throughout this work. Also the security requirements of real sample applications, representing a wide scope of applications, are studied and security functions needed to fulfill these requirements are found. Some general considerations on the placement of these functions in the layers of OSI are presented.

In chapter 4, the security functions are placed into the layer of OSI based on the criteria set in chapter 3. Mechanisms for implementing these functions are found, based on the analysis presented in the appendix.

In chapter 5, management issues associated with the secure OSI stack sketched in the previous chapter are discussed. Methods for the activation and negotiation of the use of

these security functions and mechanisms are found. Key management, which is one of the most important issues in any large, secure system, are also dealt with.

In chapter 6, implementational issues are discussed. We aim at solutions that can be implemented with reasonable modifications to the existing open systems implementations. In this chapter we have a look at some existing OSI implementations, the use of software and hardware, and existing cryptosystems that can be used in implementing the mechanisms sketched in chapters 4 and 5.

In chapter 7, the results of this work are applied to a real case. A subset of the secure OSI protocol stack is used in building a secure FTAM system, based on existing OSI implementations.

In chapter 8, a short overview of this work is given, the results are evaluated and directions for further research are pointed out.

In the appendix, security mechanisms which can be used to implement the security functions of chapter 3 are studied. Some areas of interest to this work, which are more or less tutorial in nature and are clearly outside the core of this work, are also discussed here.

2 The Basis and Scope of This Work

2.1 The Basic Model

For the purposes of this work the world is divided into four parts as depicted in figure 2-1 below:

- The communicating *application instances*, consisting of the user (or service), the application process serving it, and the part of the OSI stack dedicated to serving this application process, running on *end systems* which are assumed to be trustworthy and physically protected.

- An untrusted *network* interconnecting the end systems.

- *Trusted functionality* residing beyond the network. At least an off-line *Certification Authority* (which need not be connected to the network) and, optionally, a trusted on-line *Security Server*.

- The *outer world*, including enemies attacking the interconnected end systems, both through the network and directly at the end systems.

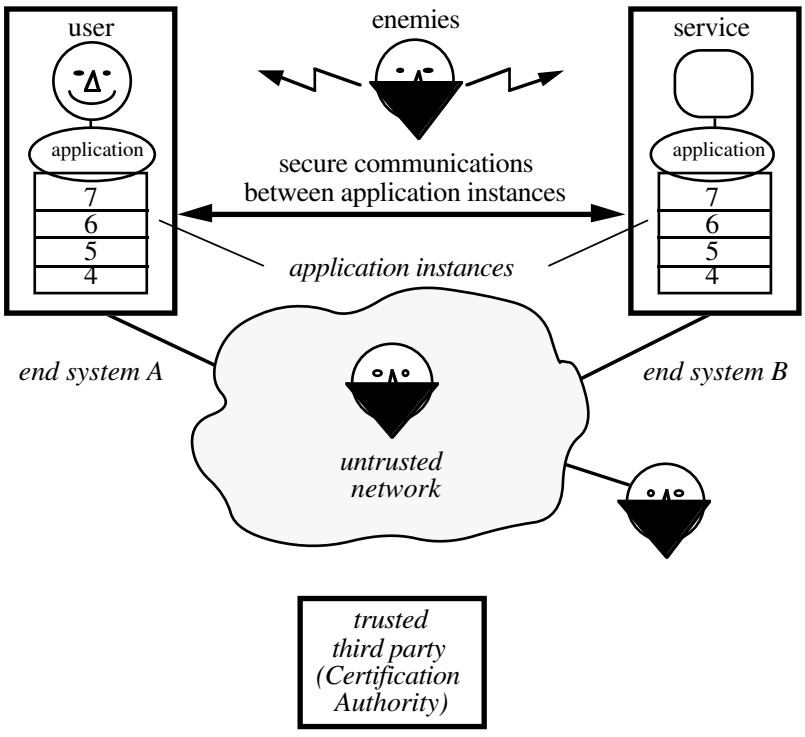

Figure 2-1: Model of the world: application instances at trusted end systems communicating securely across an untrusted network.

This work is based on the following ideas:

- The client-server model. The network is seen as interconnecting various services and their users. This model is commonly accepted as the basis of modern distributed information systems.

- Openness and globalness. It is seen that the value of various information services is directly proportional to their geographic coverage and the number of users that can reach them. It is essential that these services can be made accessible across organizational and geographic boundaries.

- The necessity of security in open systems. For these global services to be valuable in real life, it is necessary that they can be used securely and that commitments can be made electronically.

2.1.1 End Systems and Application Instances

The *end systems* are computers hosting applications and interconnected via an untrusted network (see figure 2-1). In this study, the end systems, including their local terminals and applications running on them, are considered trustworthy by their owners. However, a user does not trust other users of the same end system nor do the end systems trust one another, unless otherwise stated. Furthermore, a user does not trust a foreign end system, such as a vending machine, he is using.

Physical and local security of end systems have been widely studied and techniques have been developed for ensuring them. Lots of work has been done in the area of securing the end-systems and it will not be dealt with in this study. A short overview of work already done in this area follows.

Multi-level security models, such as the Bell-la Padula model [BP74] introducing the concept of *security classes* extended by Biba [Bib77] with *integrity classes*, have been developed for formalizing and implementing security policies. A brief summary of security policies and models is presented in appendix 4.

When this study was already written I received a preprint copy of Rainer Rueppels paper *A Formal Approach to Security Architectures* to be presented in Eurocrypt'91 [Rue90] (dated Dec. 17, 1990). This paper appears to constitute a fundamental piece of work in the area of formalizing the design and analysis of secure information systems. The formalism developed could probably be adapted to the formal analysis of security contexts and secure communications sketched in section 3.4.

In the *Trusted Computer System Evaluation Criteria* (*TCSEC*) of the US Department of Defense (DoD) [DoD85], also known as the "Orange Book", computer operating systems are divided into seven security classes (A1 through D) grouped into four divisions. The criteria and procedures for evaluating computer operating systems, covering the aspects of Security Policy, Accountability, Assurance and Documentation, are defined. However, telecommunications are not included in this evaluation. As soon as a computer is connected to an untrusted network, it in practice loses its security clearance. No security requirements or validation procedures are defined for telecommunications subsystems of computer operating systems in the Orange Book.

The "Red Book" [NCS87] extends the evaluation criteria of the Orange Book to cover computers connected to trusted networks. However, untrusted networks, which are the basis of this study, are outside the scope of the Red Book.

The European counterpart of the Orange Book is the *Information Technology Security Evaluation Criteria* (*ITSEC*) [ITS90] defining the harmonized criteria of France, Germany, the Netherlands and the United Kingdom (and probably in the future of the European Communities). In ITSEC six *assurance levels* (E1 through E6) are defined, indicating increasing degree of trust in the *Correctness* and *Effectiveness* of the security functions of a computer system. In addition, ten predefined *Functionality Classes* (F1 through F10) are specified with F1 (together with E2) corresponding to the TCSEC class C1 and F5 (together with E6) corresponding to A1. E0 corresponds to class D.

A great deal of practically oriented literature is available on the topic of secure information systems. For example, [Woo87] gives a rather comprehensive checklist of the security controls of an information system and [Woo90] gives an overview of the design principles of a secure information system.

In this study, the term *application instance* is used to mean one instance of an application consisting of the user (or service), the application process serving it, and the part of the OSI stack dedicated to serving this application process (namely the OSI layers 4 through 7). Each application instance runs on an end system and communicates with other application instances, running on other end systems, via an untrusted network. In this study, an application instance is trusted by its user, unless otherwise stated.

It is worth pointing out that the division between the end system and the network is somewhat artificial. In fact the end system is a network of circuit boards interconnected via the backplane. Similarly each ciruit board is a network of interconnected chip carriers. One chip carrier may contain several chips ponded together. The ultimate communicatings elements communicating via an untrusted network are the individual chips which can be made physically secure.

This revelation gives rise to a unified view on security where network and end systems security are no longer treated separately. By adding these lower layers to the network hierarchy depicted in figure 1-1 we can view any information system as an interconnections of chips communicating via an untrusted network. Our ultimate goal therefore is to find the basic elements of security and implement these on each chip. These thoughts were brought up in the discussion with D. Farber [Far91].

2.1.2 Network

The *network* is an internet, consisting of a number of interconnected networks, whose security cannot be guaranteed. The network may (accidentally or voluntarily) e.g. lose, duplicate, generate or change messages passing through it. When a trusted, locally secure system is connected to a network, a whole new group of threats coming in through the network needs to be taken into account.

It deserves to be noted that teleoperators are doing their best to guarantee the security of public data networks. Their cablings are physically secured and their cross-connections and active components reside in secure premises and are operated by trusted personnel. In addition, many of the network services offer security enhancements, such as closed user groups or authentication of the calling party (the A-subscriber). It is also likely, that teleoperators will increasingly often employ cryptographic techniques in their networks in the near future. However, at least for yet a long time, not all public networks can be trusted. In this study the network is always assumed to be untrustworthy.

2.1.3 Trusted Parties

In order to carry out secure communications we always have to trust someone. However, the amount of trusted parties needs to be kept at its minimum. At least one trusted *Certification Authority* (*CA*) is always needed. That is, a third party whom we can trust to certify at least the authentication parameters of ourselves and those of other parties and, optionally, to notarize agreements etc. In case of disagreement, this trusted third party can act as an impartial judge (or witness).

The minimal trusted third party is an off-line Certification Authority certifying the public keys of other parties. These certified public keys can then be used for purposes of authentication, safe distribution of session keys, ensuring the integrity and confidentiality of the transferred data, and non-repudiation. It is worth noting that certificates issued by the CA can be distributed through an untrusted channel, such as the Directory Service, without the danger of them being compromised.

Even though no trusted on-line server is actually needed (only a trusted off-line authority), the use of a trusted on-line Security Server for purposes of e.g. Key Generation

and Distribution, Notarization, Security Logging etc. can often simplify things. For example, in [BM90] a security scheme for selling the spare time of a computer to other *authorized* users is proposed, based on *coupons*, DES and a trusted *broker*.

In this study, we try to minimize the number of trusted parties. We expect each user to trust his own CA. We also assume the transitivity of trust, that is we trust the CAs that a CA trusted by us trusts. This makes it possible for the CAs to form global chains of trust.

2.1.4 The Outside World

The outside world includes *enemies* that may have access to the network or to the end systems. The three masked villains in figure 2-1 represent enemies attacking the end-systems directly, through the network (a crooked user), and from within the network (a crooked teleoperator). Of these three types of attacks, the latter two are within the scope of this study.

Possible attacks by enemies through or from within the network include, but are not limited to, the following:

- Masquerade, somebody else trying to impersonate as one of the communicating parties.

- Eavesdropping, unauthorized passive monitoring of traffic.

- Manipulation of messages between the communicating parties (e.g. changing the sums in banking transactions).

Especially nodes acting as gateways between two networks (of any kind) can intercept traffic and launch any kind of active attack towards the integrity of communications.

2.2 Communications Categories

Distributed information services can broadly be divided into two main categories by their communications requirements:

- *Connection-oriented services*, where an end-to-end connection is built between the two communicating parties. Examples of this category are *Virtual Terminal* (*VT*) [IS9040, IS9041] and *File Transfer, Access and Management* (*FTAM*) [IS8571-1,2,4].

- *Connectionless (request-reply type) services*, where a service request is generated and sent by the user and a response message is generated and sent by the service provider. For these services it is usually not worth while to build a connection at any layer and they are more naturally based on a connectionless network and and upper layer services. Examples of this category in the near future will be *Directory Access* [X.500, IS9594-1] or *Remote Database Access* (*RDA*) [IS9579-1,2]. Connectionless communications at all the layers of the OSI model are clearly the most natural basis for this type of services.

As a special case of connectionless services, *relayed (store-and-forward type) services*, where no real-time end-to-end communications take place, have to be considered. Examples of this category are the Message Handling System [X.400] or file transfer through an intermediate file store.

The first category is OSI-wise the purest in the sense that the OSI reference model and most of its associated standards were first designed with this kind of communications in mind and have later on been extended towards the use of connectionless services.

The second category is of increasing importance because it fits well in with the Client-Server Architecture, which seems to be the current trend in information systems. With the advent of the Directory Service and Distributed Database Management, the OSI standards have been extended towards this direction, e.g. with the definitions of a connectionless transport service [IS8072/A1] and protocol [IS8602], session service [IS8326/A3] and protocol [IS9549], presentation service [IS8822/A1] and protocol [IS9576], and ACSE service [IS8649/A2] and protocol [IS10035]. [IS9545/A1] deals with connectionless operation of the application layer. It is likely, that in the near future both the use of the Directory and Remote Database Management will be based on connectionless communications services. At the moment they both still run on the Remote Operations Service (ROS) of the application layer and connection-oriented transport service.

Relayed communications are also increasing in importance, mainly because of the Message Handling System (MHS) [X.400] which is among the first internationally standardized open systems of commercial importance. However, from the OSI perspective, this kind of service is not pure peer-to-peer communication. In the OSI sense, one hop on the path of a message in the MHS, from User Agent (UA) or Message Store (MS) to Message Transfer Agent (MTA), from MTA to MTA, or from MTA to UA or MS, forms an instance of end-to-end communications in the OSI sense. Between the hops, the message is raised into the domain of applications, stored and later passed on. In MHS the lowest layer of real end-to-end communications is the UA-to-UA protocol [X.400] (also known as P2, see figure 2-2).

Among the most important applications of MHS is Electronic Data Interchange (EDI). Basically EDI means the computer-to-computer exchange of information relating to trade activities [TED89]. The information is in the form of EDI messages, which are structured electronic messages for administrative or commercial purposes. EDI strives to automate the bulk exchange of routine data, such as orders and invoices, sent between organizations. The use of MHS with EDI is discussed e.g. in [Gen90]. A concise introduction to EDI, including legal aspects etc., can be found in [Com89]. The TEDIS Program [TED90a] gives an overview of current European EDI activities. In [TED90b] a specific and detailed scheme for signing EDIFACT messages is proposed, including technical details and references to real cryptosystems to be employed.

In connection with MHS, the secure general-purpose OSI protocol stack (drafted in chapter 4) can be used to enhance security on connections between the agents (UAs, MSes and MTAs) of a MHS but these measures alone cannot guarantee end-to-end security because the intermediate nodes cannot be trusted.

This is analogous to the situation with true end-to-end services, such as FTAM, where end-to-end security has to be implemented at layer 4 (the lowest layer with end-to-end significance) or above and any security measures at layers 1 through 3 can only enhance the security of the total service but not alone guarantee it. In MHS true end-to-end security can only be achieved at the level of P2 (see figure 2-2) or higher (within the application process).

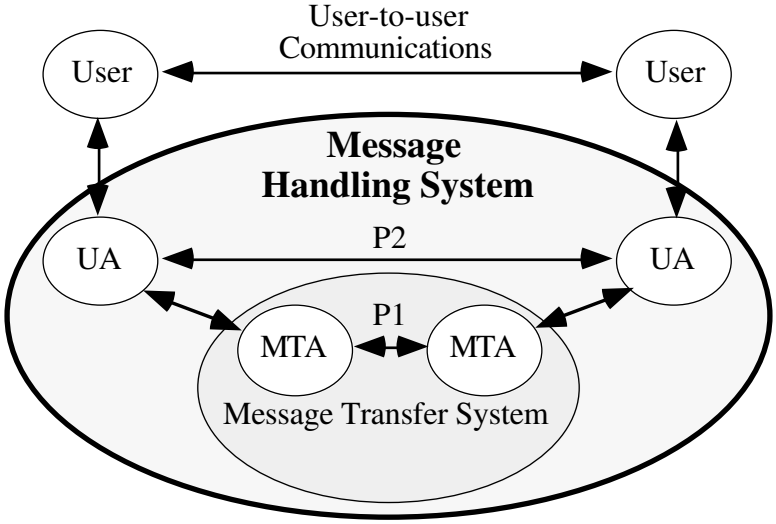

Figure 2-2: Message Handling System.

In many respects the service provided by an MHS is comparable to that provided by an ISO Internet. From the user's point of view both of these services transfer datagrams across a network of interconnected networks the main differences between them being

that MHS operates at the application layer whereas Internet operates at the network layer and that the Internet is real-time whereas MHS is not. In an internet the delays are typically in the order of milliseconds whereas MHS causes significantly longer delays, typically in the order of minutes or even hours.

Therefore, with MHS (and other relayed services) we have to include all the information needed for Integrity, Confidentiality, and Authentication (or Non-Repudiation of Origin) within the message. Only Non-repudiation of Delivery should be done with a separate acknowledgement message. With connectionless end-to-end services, such as the directory, we can exchange messages in real-time and use, for example, a two-way handshake mechanism.

In this work the analysis of MHS (and other relayed services) is restricted to the layers of OSI and the application part of MHS is only briefly dealt with. The general-purpose secure OSI stack drafted in chapter 4 can be used with MHS and it will make it possible for the communicating MHS agents to mutually authenticate and then communicate in a secure manner. However, since the intermediate agents cannot in general be trusted, true security in MHS requires Data Origin Authentication, Integrity and Confidentiality to be implemented within the message being transferred. These arrangements are a research area in their own right and beyond the scope of this work.

Irrespective of the communications category, a basic requirement in secure communications is always the support for a common *Security Context* (defined in 3.1) between the communicating parties. This includes things such as a mutual understanding of the encryption functions, mechanisms and keys used between the communicating parties at a given point in time at all the of layers OSI. Prior agreements as well as various management and negotiation mechanisms can be used in establishing this Security Context. In real-time communications we can establish the security context by means of negotiation between the communicating parties (and possibly others) whereas in relayed services the security context can only be established by means of prior agreement, management, and information included in the one-way message.

2.3 The Scope of This Study

This study deals with secure communications between trusted end-systems through an untrusted network (the arrow between the end-systems in figure 2-1).

In order to protect against the threats posed by the untrusted network, secure protocols between these end systems have to be devised. It is also possible, and sometimes even necessary, to involve a trusted third party called the security server in these transactions.

This work concentrates on the threats coming in through the network. Local threats, such as *insider threats*, *Trojan horses* etc. are beyond the scope of this study. The communicating end systems are assumed to be secure and physically protected. Referring to the restrictions above, such services as *Data Flow Confidentiality*, *Routing Control* and protection against *Denial of Service* are also beyond the scope here (see [IS7498-2] for definitions of these threats).

In this study we do not concentrate on cryptography, e.g. the mathematical properties of real cryptographic mechanisms are not analyzed. The requirements for various mechanisms (such as *symmetric* and *asymmetric cryptosystems*, *cryptosums* and *hash functions*, and *digital signature schemes*) to be used in implementing various security functions are stated and mechanisms complying with these requirements are assumed and used as basic building blocks of secure open systems. For each such abstract mechanism an existing mechanism (such as DES or RSA) which is currently believed to fulfill these requirements is found (see Appendix 1). However, this work is not tied to any existing mechanisms and they can be replaced as needed.

It is expected that in the future various symmetric and asymmetric cryptosystems and *zero-knowledge schemes* will be designed and broken in a continuous struggle between *cryptographers* and *cryptanalysts* (code makers and code breakers). This study being independent of any specific cryptosystems, new cryptosystems can be adopted with only minimal modifications to the security profiles drawn up here.

This work deals with the architectural aspects of open systems security. Based on the OSI reference model and its associated security framework, answers are sought to questions such as what type and level of security is needed (and feasible) with various real information services, what combination of security functions will provide the overall level of security needed with each service, and what layers of the OSI reference model should these functions be placed in.

Mechanisms used to realize these functions as well as implementational issues, such as the use of hardware and software in implementing these mechanisms and the implications of the chosen software and hardware architecture on the overall security of the system, are studied. To facilitate this work, a semiformal security model is developed and used to analyze the security functions derived from actual security requirements of real open systems.

2.4 Approach to the Goals

In this work, the security requirements of the three categories of information services listed in 2.2 are analyzed. FTAM and Virtual Terminal are used as examples of connection-oriented services, the Directory Service and Remote Database Access as examples of connectionless real-time services, and the Message Handling System as an example of relayed services. Their required types and levels of security are defined, the combinations of security functions to achieve the required overall security levels are found, these functions are placed into the layers of OSI, and mechanisms are specified to implement them (including cryptosystems and extensions to current protocols).

Based on this analysis, a general purpose OSI protocol stack with built-in security is drawn up. This stack can be used with most connection-oriented and connectionless end-to-end services and it can also be used to provide "hop-by-hop security" in relayed services such as MHS. The required management services and mechanisms for this protocol stack are sketched.

For FTAM a complete security profile is drawn up and the ideas presented in this work are tested in practise with a limited working prototype implementation of a secure FTAM service. The solutions are evaluated against the initial security requirements, possible enhancements are sketched and directions for further work are pointed out.

3 On Security Requirements and Functions

Various information services have different security requirements which can be met by the five security services listed in the OSI Security Architecture [IS7498-2], namely: *Authentication*, *Integrity*, *Confidentiality*, *Non-Repudiation* and *Access Control.* The OSI Security Architecture lists these five services and a number of subclasses of each, such as *Connection* or *Connectionless Integrity With* or *Without Recovery*, with little insight to their significance or the relations between them. Here we shall analyze the interdependencies between these services and make some general observations. Also some sample services are analyzed with regard to their specific security requirements.

File Transfer, Access and Management (*FTAM*), the *Directory*, and *Message Handling System* (*MHS*) were chosen as the sample services for the following reasons:

- All of these services are already rather mature (implementations do exist) and seem to be useful in real life in the near future.

- These three services represent the three categories of services described in 2.2 and a closer analysis of them should reveal something applicable to virtually any application.

Even though the term *Security Service* is used in the OSI Security Architecture, it does not mean a service in the normal OSI sense of the word. In the OSI Reference Model the term *(N)-service* is defined as: "A capability of the (N)-layer and the layers beneath it, which is provided to (N+1)-entities at the boundary between the (N)-layer and the (N+1)-layer" at an *(N)-service-access-point* [X.200] by means of (conceptual) *(N)-service-primitives* [X.210].

However, in the Security Architecture addendum of OSI [IS7498-2] *Security Service* means a more abstract kind of functionality provided by the entire OSI system. An OSI Security Service, in the sense it is being used in the OSI Security Architecture, need not show at all in the service interface of any layer. The use of a Security Service, such as Confidentiality, may be dictated by a *Security Policy* and enforced by *Systems Management* without the user knowing anything about this.

Therefore, it would be more appropriate to talk about *Security Functions* when we mean security as a part of the overall quality of service provided by the OSI system and reserve the word *Security Service* to mean services offered at the boundary between two OSI layer by means of service primitives. This practise is followed hereinafter in this study.

Definitions of some terms used in this study:

- *(N) Security Service:* a security-related capability of the (N)-layer and the layers beneath it, provided to (N+1)-entities at the boundary between the (N)-layer and the (N+1)-layer by means of service primitives.

- *(N) Security Function:* a security-related function, enhancing the quality of service provided by the (N)-layer, controlled by the control part of (N)-entity and activated at the request of the user of the (N)-service or by systems management.

- *(N) Security Mechanism:* a mechanism at the (N)-layer realizing (a part of) an (N) security function.

3.1 A Security Model

A layer entity A_N (entity in system A at layer N) is depicted in figure 3-1, below. The entity is divided into the following parts:

- *Control Part* realizing the protocol logic and controlling the use of functions and variables.

- *Mechanisms* controlled by the control part, operating on the variables and realizing functions.

- *Variables* most of which are local to each entity instance.

The Control Part is usually modelled as a state machine. In the C-VOPS environment an Extended Finite State Automaton (EFSA) is used to model and implement the Control Part [Kar87].

Mechanisms include three major groups, each realizing one interface of the (N) layer entity:

- *Upper Interface Mechanisms*, realizing the interface to layer N+1 (the (N) Service Interface).

- *Lower Interface Mechanisms*, realizing the interface to layer N-1 (the (N-1) Service Interface).

- *Protocol Mechanisms*, realizing the protocol interface to the peer entity.

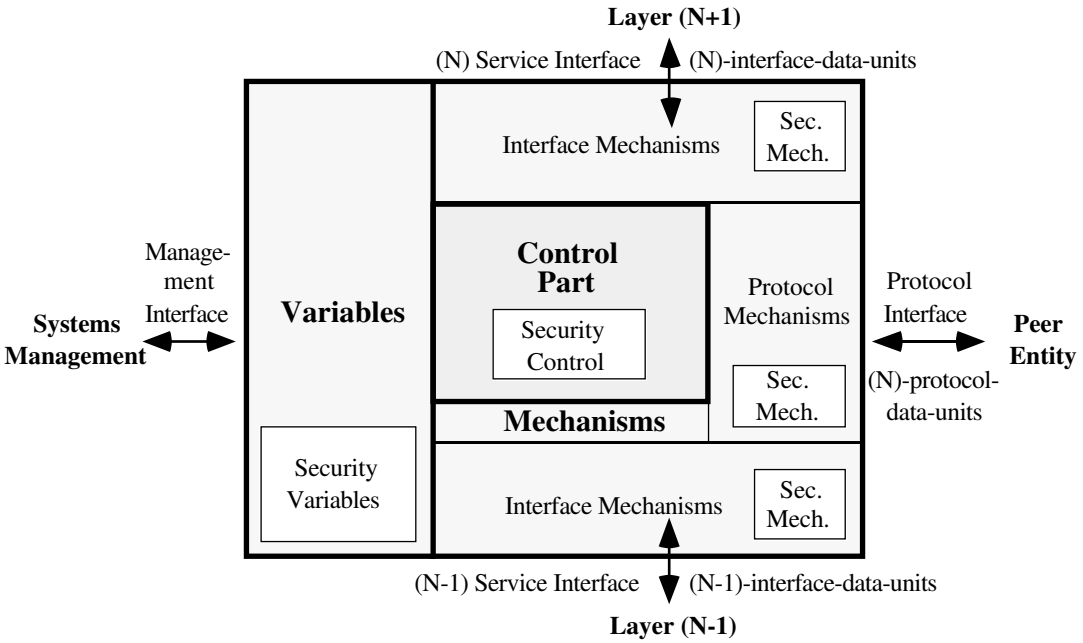

Figure 3-1: OSI layer N entity security architecture.

Formally, layer entity A_N can be modelled as a triplet: (C_{AN}, V_{AN}, M_A), where C_{AN} is the control logic, V_{AN} is the set of variables, and M_{AN} is the set of mechanisms of A_N. The set of variables of layer entity A_N is the union of the set of *Protocol Variables* of entity A_N and the set of *Security Variables* of entity A_N:

$$V_{AN} = V_{AN,P} \cup V_{AN,S}$$

The set of security variables of application instance A is the union of the sets of security variables of all of its layers 4 through 7 and the security variables of the application:

$$V_{A,S} = (\cup_{i=4,7} V_{Ai,S}) \cup V_{AA,S}$$

Similarly, the set of mechanisms of layer entity A_N is the union of the set of *Protocol Mechanisms*, the set of *Interface Mechanisms*, the set of *Security Mechanisms*, and the set of other mechanisms of A_N:

$$M_{AN} = M_{AN,P} \cup M_{AN,I} \cup M_{AN,S} \cup M_{AN,O}.$$

Security variables include key values and information controlling the selection and use of various security mechanisms. The values of security variables can be set by systems management (as dictated by the security policy) or as a result of negotiation at an upper layer of OSI, or the control part of the layer entity (as a result of negotiation or key exchange within layer N).

Layer N Security Context between the layer entity instances $A_{N,i}$ and $B_{N,j}$ can now be formally defined as:

$$SC_{AN,i,BN,j} = (V_{AN,i,S},\ V_{BN,j,S})$$

To be usable in real communications, a security context has to include a lot of things left implicit in this definition. For example, we need to have a common understanding of the cryptographic mechanisms employed, their modes of operation, the initial default security context etc. Many of these aspects belong to the domain of security management. In this study we shall assume that the security related variables contain not only such information as key values etc. but also control information used to select and activate the appropriate cryptographic mechanisms. Further elaboration of the concept of security context is left outside the scope of this study.

Because in this study we are interested in secure communications between entire *application instances* as defined in section 2.1.2 (see figure 2-1), not in individual layers of the OSI model, we also need to define the security context between application instances. We therefore define the security context between application instances A and B (at all the layers 4 through 7 and the application) as:

$$SC_{A,B} = (V_{A,S},\ V_{B,S})$$

The respective systems managements, in cooperation with the Control Parts of all the OSI layers 4 through 7, are responsible for negotiating the Security Context between two application instances, using the Security Mechanisms to perform the selected Security Functions within this Security Context, and releasing the Security Context at the end of communications. The life span of a security context can now be divided into three consecutive phases: *Set-up*, *Use* and *Termination.* More formally, we shall write: *Set-up*($SC_{A,B}$) and *Terminate*($SC_{A,B}$) for these two operations.

A Security Context can be set up by means of prior agreement, management and negotiation. It is possible to negotiate several Security Contexts, label them (e.g. with small integers) and manage them in much the same way as Presentation Contexts are managed at the Presentation Layer.

Security functions can be *Autonomous*, that is they only show to the security profile and need not be activated by the user (they are either always active or are activated by the systems management guided by the *Security Policy*), or *Activatable* (security services), meaning that their use is negotiated and controlled by the service user with service primitives and their parameters.

A *layer entity* is a static type definition of *layer entity instances* which together constitute a class. The layer entity A_N is like a program which can be invoked many times over whereas the layer entity instance $A_{N,i}$ is a dynamic invocation of such a type, like a process executing the program. In OSI both types and instances of layer entities carry *distinguishable names*. Usually it is clear from the context when we are referring to types and when instances of layer entities. From here on, we shall talk about an *entity* meaning either an entity *class* or an entity *instance*. When it is not clear from the context which we mean, the distinction is made explicitly.

When establishing an *(N)-connection*, the *(N)-user* issues and *(N)-connect-request* ((N)-CR) primitive to an *(N)-service-access-point* ((N)SAP) of the *(N)-service-interface*. This service request typically invokes a new instance of the (N)-entity, dedicated to serving this one instance of communications, which immediately issues an *(N)-connection-request* ((N)CR) *Protocol Data Unit* (PDU) to its peer entity. At the other end system, the (N)CR PDU invokes a new instance of the (N)-entity, also dedicated to serving this one instance of communications. All PDUs on this connection now flow between these two peer entity instances (called *correspondent entities*) and service primitives of this connection at the *connection-end-points* (CEPs), distinguished by their *connection-end-point-identifiers* (CEPIs), are directed to and issued by these entity instances.

Communications can only occur between layer entity *instances* and, therefore, e.g. connections, associations and security contexts cannot exist between entity types but always between pairs of layer entity instances. In a real open system, each instance of the same layer entity type executes independently of all other instances. This implies, for example, that they have their own sets of variable instances (local variables).

While there is nothing in the OSI reference model explicitly denying us from establishing connections between specific instances of entities (if we know their names), we usually only are interested in entity types when opening a connection. In fact, the current OSI protocols do not allow us to address individual entity instances. After the connection is established, we never need to explicitly refer to entity instances, but they are associated with the connection-end-points of this connection at the respective (N)-service-access-points.

With connectionless services the situation is essentially the same – we only need to know the address of the other party and issue a service request to the appropriate (N)SAP.

Here again we suffer from the connection-oriented tradition of the OSI reference model. It would be more appropriate to adopt the concept of *association*, originally born for use in the application layer (*application association*), which does not make a distinction

between connection-oriented and connectionless communications. An *(N)-association* is established between two entity instances at layer N in order to facilitate their mutual communication. (N)-association can be supported by an (N-1)-connection or connectionless (N-1)-service. It is evident, that the OSI reference model is evolving towards equally supporting connection-oriented and connectionless communications. One example of this development is the connectionless *Association Control Service Element* (*ACSE*) of the application layer defined in [IS8649/A2] and [IS10035].

3.2 Security Requirements

Here the requirements are seen mainly from the user's point of view. When using a connection-oriented information service across the network, the user may need to know one or more of the following:

- That the identity of the other party is that claimed and that it remains the same throughout the session.

- That nobody else can listen to (and understand) the session.

- That nobody can undetected delete from, change, or add to the information transferred.

- That commitments made during the session can, beyond reasonable doubt, afterwards be proved to an impartial judge.

Similarly, the service provider may need to know the same plus the following:

- That nobody except the legitimate users can access the service.

- That the service provider can, if necessary, prove to an impartial judge that the user actually has used the services he is charged for.

In OSI terminology [IS7498-2], the five security functions needed to provide for the five first requirements stated above are called *Peer Entity Authentication*, *Connection Confidentiality*, *Connection Integrity*, *Non-repudiation*, and *Access Control*, respectively. In the case of connection-oriented communications, a Security Context can be set-up when establishing the connection and remain in effect throughout the session.

With services not based on an end-to-end connection between the communicating parties, the situation is somewhat different. With these services, connectionless versions of the above functions are needed, such as *Data Origin Authentication*, *Connectionless Confidentiality*, and *Connectionless Integrity*. Also, since there is no

connection, there cannot be a Security Context associated with a connection but rather a Security Context has to be associated with each message or request-reply pair.

3.2.1 Security Requirements of FTAM

File Transfer, Access and Management (*FTAM*) [IS8571-1] is a typical end-to-end service, where layers 4 through 7 communicate in real time, on-line from one end system to the other. FTAM is currently the most mature connection-oriented service but results obtained here should be rather easily applicable to such services as Virtual Terminal (VT) [IS9040], Job Transfer and Manipulation (JTM) [IS8831, IS8832], and Reliable Transfer (RTS) [IS9066-1,2, X.218, X.228].

In FTAM the concept of a file store is generalized and the real file systems residing on various host systems are mapped into a global Virtual File Store [IS8571-2] where files can be accessed independently of their physical locations.

In real file systems we need to give individual users and user groups various access rights to each file. Usually, we can specify for each file what users and user groups are allowed to read, write, execute or delete it. The local operating system is responsible for enforcing these restrictions on the user of files. Local access control is based on user IDs and the user authentication scheme used by the operating system. The access control mechanism can be implemented by associating an access control list with each file or by assigning appropriate capabilities to each legitimate user.

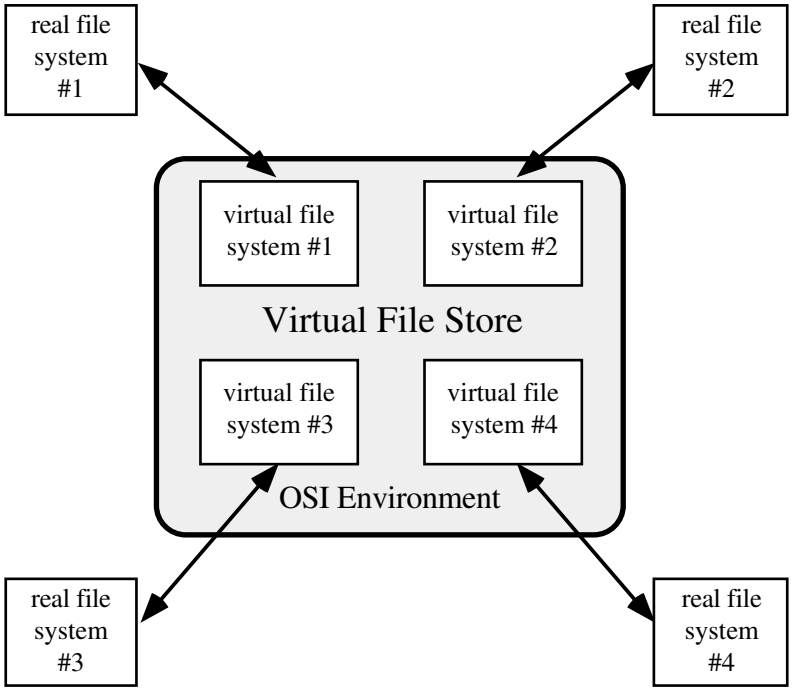

Figure 3-2: The FTAM Virtual File Store.

With the FTAM Virtual *File Store*, we have the same threats as those with real file systems, and some more. Because the virtual file store is distributed globally it reaches across organizational, administrative, social and cultural boundaries. Not all the parts of the system are controlled by the same organization or even by the same laws and ethics. This brings with it new worries that did not exist with a local real file store.

With the virtual file store we need the same kind of security as in real file systems with sufficient granularity for being able to enforce these restrictions on individual users, groups of users, and types of use. However, in a global open environment the number of users who could attack the system is much greater and even in legitimate use the often confidential data has to be transferred across an untrusted network.

While in a local environment we can normally rely on the user IDs as a basis for access control, in a distributed environment we need strong *Peer Entity Authentication* (for a definition of Strong Authentication see 4.3). Furthermore, we need to make sure that the peer entity authenticated at the beginning of a session remains the same throughout the session.

This means tieing *Connection Integrity* with *Authentication* in such a way that a change of the peer entity at any time will be detected. The integrity function needs to make sure that the whole data stream is intact, that is no part of it has been changed, omitted, duplicated or misplaced. With FTAM (and VT) the purpose of the integrity function is not only to protect the integrity of the data being transferred but also to extend the authentication of the peer entity from a one-time operation to cover the whole session.

If the data being accessed is confidential, then a Confidentiality function should be invoked. With services such as FTAM and VT non-repudiation is usually not an important issue, undisputability being a property of the data itself and independent of the way the data is transferred.

Based on the above discussion, it can be claimed that the most important security functions needed with FTAM (and VT) are (in this order): *Peer Entity Authentication*, *Connection Integrity*, and *Connection Confidentiality*.

3.2.2 Security Requirements of the Directory

The same Authentication, Access Control, Confidentiality and Integrity functions that are needed with FTAM and VT are also needed when using the Directory Service. While the Directory plays an important role in most OSI applications as the means of finding various services and distributing authentication information [X.500, X.509, IS9594-1,8], it is also itself an OSI application that needs to be secured.

Secure use of the Directory is important because not necessarily all the information stored in the Directory is public and some of it needs to be protected against modification. The main distinction between the Directory and FTAM is that the use of the Directory is most naturally based on connectionless communication service, even though it currently still is running on the *Remote Operations Service Element* (*ROSE*) [IS9072-1,2, X.219, X.229] and connection-oriented protocols from the transport layer up. A closer analysis of the Directory should reveal something applicable to, at least, *Remote Database Access* (*RDA*) [IS9579-1,2], *Job Transfer and Manipulation* (*JTM*) [IS8831, IS8832], and *Network Management* (for an overview and some more detailed information on systems management, see e.g. [IS10040, CPW89, IS9595, IS9596]).

One of the main obstacles for a wide acceptance of a global distributed Directory Service is that most companies do not want to reveal the details of their internal organization, personnel etc. to their competitors. This would be the same as making the company's internal telephone catalogue public and, furthermore, distributing it in an electronic form ready for duplication and automatic processing.

What is needed in a typical large company is a secure directory server, being under the control of the company but forming a part of the global directory system, that gives each user secure access to precisely the information that he is entitled to access. The secure use of the directory service appears to be a necessary prerequisite for the proliferation of the whole service.

Data Origin Authentication needs to be applied to each message in order to assure the directory server that the request came from the user claimed and the user that the response actually came from the directory server claimed. This gives the server a firm basis for applying access control and the user an idea of the degree of confidence that he can have in the correctness of the information received.

As with virtually any application, data origin authentication would be worthless without an associated data integrity function. Integrity together with Authentication gives the user and server an assurance of that the request or reply received has not been tampered with (e.g. by an enemy within the network).

The Confidentiality Function is needed when critical data stored within the directory is accessed. Usually this is not as important an issue as Authentication and Integrity are because it is much more difficult and less efficient for an enemy to wait for the piece of confidential information that he wants to get to pass by than it would be to ask for it directly from the directory. In any case, a data confidentiality function is needed when accessing the most critical pieces of information stored in the directory.

Based on the above discussion, it can be claimed that when accessing the Directory, or a remote database, the most important security functions are the following: *Data Origin Authentication*, *Connectionless Integrity*, and *Connectionless Confidentiality*. These are simply the connectionless versions of the functions needed with FTAM, JTM and VT.

3.2.3 Security Requirements of the Message Handling System

The by far most significant relayed information service today is the CCITT X.400 *Message Handling System* (*MHS*) or, in ISO terms, the *Message Oriented Text Interchange System* (*MOTIS*). One of the most important applications of MHS is *Electronic Data Interchange* (*EDI*) aiming to replace a large part of inter-corporate documents, currently transferred on paper, with electronic trade documents.

Among the security threats associated with MHS are the following:

- A recipient making a false claim of the origin of a message.
- The sender falsely denying the submission of a message.
- Somebody falsely acknowledging the receipt of a message.
- The recipient falsely denying the receipt of a message.
- Somebody impersonating as another user of the MTS (a UA impersonating as another UA to an MTA).
- An MTA impersonating as another MTA (to the user or to another MTA).

Security of the MHS can be divided into two parts:

- *Hop-by-hop security*, focusing on communications between two consecutive agents of the MHS.
- *End-to-end security*, which mainly belongs to the domain of applications and is therefore beyond the scopes of both OSI and this study.

While an individual user is mainly interested in the end-to-end security of his message (the first four threats listed above), a company using or providing the MHS service should be interested in both aspects of MHS security (including the last two security threats listed above).

Hop-by-hop security of MHS is important for several reasons:

- In order to guarantee that messages are ultimately delivered to their right recipients it is important that an MHS agent shall not forward messages to anybody but the right peer agent.

- Even if a message is encrypted it should not be exposed to an untrusted agent. An enemy with access to the message stream could, for example, easily perform extensive traffic analysis based on the addresses on the cleartext envelopes or make a serious attempt to break the encrypted messages assumed to be important based on the envelope information.

- For a long time yet, most messages will be unencrypted. While these messages may not be overly critical they should by no means be entrusted to just anybody or subjected to easy alteration by the network. While the interacting MHS agents are not all trustworthy they still pose a very small security risk compares with that posed by all the other parties connected to the same network.

Currently, there is a lot of ongoing activity in the area of the security of MHS and the latest blue book recommendations ([X.400] and others) include several security features. The work is still far from mature. For a concise critical study on X.400 security see e.g. [Mit90].

The security requirements of the connection between two agents of an MHS are to a great extent the same as those of FTAM and VT. First the two communicating agents need to perform strong mutual authentication. Just like with FTAM, this authentication can be extended by employing an integrity function closely coupled with authentication this function also protects the transferred message stream from modifications. A confidentiality function can also be employed to protect against revealing the message flow or contents of individual, unencrypted messages. Also with MHS access control is left to the application and hop-by-hop non-repudiation is usually of little interest (except at the boundaries between operators of interconnected MHS services).

End-to-end security in MHS is concerned with individual messages transferred by the MHS. Each of these messages has associated with it a security context of its own.

Especially in EDI, but also in the exchange of free format documents, instead of data origin authentication the stronger function of non-repudiation of origin is usually required. In order to make electronic commitments it is not enough to know the identity of the other party but we also need to be able to show this to an impartial judge in case of dispute.

As always, non-repudiation is impossible without integrity. Therefore integrity is implied by non-repudiation. Integrity is also often required when non-repudiation is not.

End-to-end confidentiality is always required with truly confidential messages, since not all of the MHS agents handling the message along its way can be trusted.

3.2.4 Summary of the Security Requirements

Basically, the security functions needed by the two main categories of OSI services (*connection-oriented* and *connectionless* services) appear to be approximately the same, the main distinction being that in the former case the security context applies to the entire connection whereas in the latter case each message (or each request-reply pair of messages) has a security context of its own and that connection-oriented and connectionless versions of the security functions (respectively) are needed. Also the security requirements of the third category (*relayed services*) on each hop are approximately the same as the security requirements of connection-oriented services.

This observation seems to support the view that security, indeed, should be an integral part of the communications service provided by the OSI system. It also shows that we should look for ways of unifying the connection-oriented and connectionless versions of security functions. True end-to-end security in relayed services falls mainly outside the scope of this study.

3.3 Security Functions

Even though it is difficult to prioritize the OSI security functions, as their priorities depend on the specific user, type of use and application, some general conclusions, based on the above discussion, can be made:

- With the connectivity offered by today's integrating networks, reliable *authentication* appears to be the most urgent need. With current technologies, such as the use of passwords on sessions and reserved fields in E-mail messages, it is very easy to e.g. log in a service or send E-mail under a false identity.

- *Data integrity* is of paramount interest in virtually any information service. All essential data has to be secured against both accidental and premeditated tampering. Even data that is not valuable for an outsider can be manipulated, systematically or at random, by e.g. a competitor or a casual hacker.

- *Data confidentiality* is usually less critical an issue than data integrity. Most of the data transferred across a network is not very interesting for an outsider and usually the sheer bulk of it makes it difficult to find the relevant pieces of information. However, a part of the information transferred through the network is critical and needs to be encrypted. Among the most critical pieces of information are encryption keys distributed through the network. Sometimes entire documents, such as offers and sales reports, need to be protected. Also bank transactions are usually confidential.

- *Non-repudiation* is a very difficult area because of the legal aspects involved. Electronic commitments are also to a great extent application-dependent. However, a *Generic Non-repudiation Function*, offering the service of non-repudiation of an Application Protocol Data Unit without any regard to the semantics of the information transferred, can be included in the service provided by the OSI, as proposed in section 3.5.

- *Access control* is a "function" belonging mainly to the domains of applications and management. Based on reliable authentication, it is relatively straightforward to check the privileges of the user and allow him the appropriate access to the service. Access control only is a service from the information service provider's point of view – as seen by the user it is rather a restriction. Access control can also be applied at various layers of OSI, as pointed out in section 4.5.2.

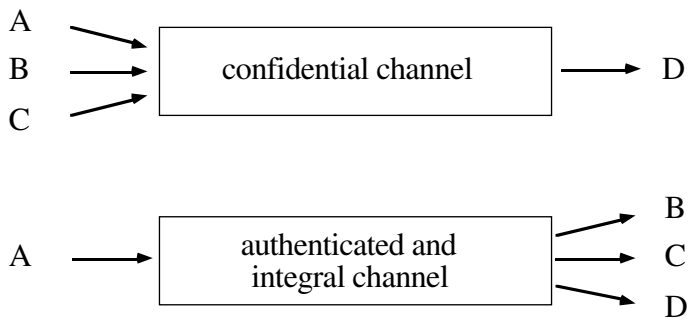

Figure 3-3: Confidential and Integral channels [Rue91].

Figure 3-3 Illustrates the functions of confidentiality, authenticity and integrity. A confidential channel is one that anybody (in this case A, B and C) can write into but only the legitimate recipient (D) can read from. An authenticated and integral channel is one that only the legitimate sender (A) can write into but anybody (B, C or D) can read from. These channels can be either physically protected (e.g. optical transmissions systems with just one receiver or transmitter) or cryptographic channels. They can also be formalized.

[Sim84] provides the basis of systematic study of authenticity. The topic is further elaborated on e.g. in [Sim88]. OSI security standards now include a fairly complete Authentication Framework [IS10181-2]. The corresponding other frameworks (the Integrity Framework [ISO90d] and the Non-repudiation Framework [ISO90e]) are still rather early drafts.

For example Integrity alone is useless, it must always be combined with Authentication to be of any real value. If we cannot be sure of the identity of the other party, it is of little value to us to know that the data has not been tampered with. Similarly, Authentication is of little value without Integrity. If we do not know that the data is intact it is not worth much for us to know who sent it.

It is easy to annul the effect of either of these two functions unless they are combined. Consider an active attack against the security of communication between two parties A and B launched by enemy X, connected between A and B and capable of intercepting all traffic between these two parties. X can be e.g. an untrusted router between two networks.

Suppose the enemy X intercepts an authenticated message from A to B. If no integrity function is provided X can now change the information contents of the message while leaving the authentication information intact. B now checks the authentication information and assumes that the message came from A when it actually did come from X.

Also, in case of connection-oriented communication, X can let the authentication exchange between A and B pass through unmodified. When the real information is exchanged, X can intercept these messages and modify them as it pleases, if no integrity function is provided.

Similarly, if only integrity but no authentication is provided X can intercept the messages and send them on as his own. B now believes that it got the information from X when it actually came from A. If this information happens to be, for example, a patent application the consequences can be serious.

In order to protect against these attacks authentication and integrity should always be provided together. During the authentication exchange a common security context is set up between the mutually authenticated parties in such a way as to keep it secret from everybody else. This means exchanging confidential information (pair-wise key values), included in the security context, as a part of the authentication exchange. The authentication exchange is performed in one security context, that is one including the certified public key pairs of the authenticating parties. In this security context, another security context, lasting for the duration of the connection, is set up and can then be used in subsequent communications. Because only the two mutually authenticated

parties know this security context, communications in it provide not only integrity and confidentiality but also continued mutual authentication.

A secure session (e.g. between an FTAM user and his server or between a virtual terminal and a host computer) should proceed as follows:

- When opening the session mutual strong authentication takes place. It is essential that a fresh *pairwise* (that is one shared by only the two communicating parties) session key be securely exchanged during this procedure.

- This session key can be used in subsequent communications for purposes of continued authentication as well as data integrity and confidentiality in a manner explained in more detail in the appendix.

- Finally, the session is gracefully closed in such a way that no data can undetected be omitted from the end of the session.

- Optionally, a digital signature scheme can be employed if parts of the session include commitments that may need to be verified (by an impartial judge) later.

There are a number of security contexts that can be involved in this process. Firstly, there is the security context between the user and his CA, where the user knows the public key of his CA and can check the certificates of the public keys of other parties. Secondly, there is the security context between the two parties knowing each other's public keys, certified by the CAs, and therefore being able to mutually authenticate and exchange information confidentially and retaining its integrity. Thirdly, there is the security context established during the authentication exchange, to be used on the newly established connection. Lastly, another security context, the union of the second and third context, can be used for making confidential commitments on this connection by using the second context for signing and the third context for ensuring the confidentiality of the messages exchanged. The security contexts employed are illustrated in figure 3-4 below.

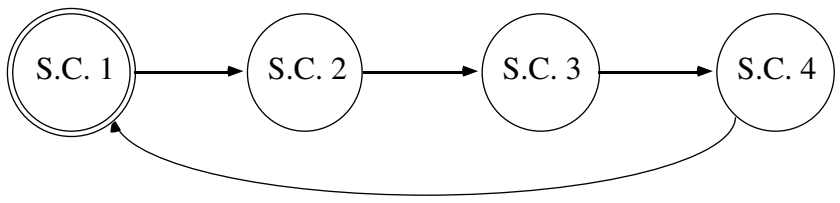

Figure 3-4: Switching between security contexts.

The purpose of the approach proposed above is to keep things simple. The initial security context is always required in order to be able to perform strong mutual authentication and securely agree on the security context to be used on the newly established connection. For purposes of continued authentication, as well as integrity and confidentiality, the third security context, based on the agreed session key and symmetric cryptography, is much more efficient than the second security context used in the authentication exchange. If commitments must be made during the session it is necessary to employ a digital signature scheme in an appropriate security context. Rather than establishing a new security context for this purpose, we can use the second security context instead. This can be done either by employing a security context which is the union of the second and third context (as described above) or by alternating between these two contexts (as illustrated in figure 3-4).

In connectionless communication each Data Unit has to carry with it all the information needed for Data Origin Authentication and Data Integrity. With connection-oriented communication the same effect is achieved by securely exchanging a secret session key in connection with Peer Entity Authentication. This key can then be used to guarantee continued authentication and the integrity of each data unit, as well as the whole data stream, during subsequent communication. This ties the Confidentiality function with Authentication and Integrity.

While Integrity and Confidentiality are orthogonal functions they are usually most naturally implemented by using the same mechanism for both.

Authentication and Non-Repudiation are not at all orthogonal but the former is a weaker version of the latter. With Authentication we *know* who the other party is, with Non-repudiation we can also *prove* this to an impartial judge.

Access Control is closely coupled with Authentication. We cannot apply Access Control without knowing who the other party is. Access Control is not a function in the same sense as the other four are. The originating user can ask for the invocation of the other functions when opening the communication whereas Access Control is imposed on the user by the communication service provider or the responding party. From the user's point-of-view Access Control is a restriction rather than a service.

Based on the above discussion, it can be stated that the security functions listed in the OSI Security Architecture are an ad-hoc enumeration of security-related issues rather than a well thought-out list of orthogonal elements of security. The three real security functions appear to be *Authentication*, *Integrity* and *Confidentiality* while Non-repudiation can be viewed as a stronger version of Authentication and Access Control really should be left to systems management and various applications, based on reliable authentication of the service user (and his end-system).

The orthogonality of the three basic security functions implies that it is generally a good practice to use independent key values for different purposes. For example, the public key pair used for authentication should be separate from the key pair used for confidentiality, even though these two functions often are based on the same cryptographic mechanism.

As a summary it can be stated that the three most important security functions of OSI are Authentication, Integrity and Confidentiality, in this order, and that these three should be provided by the OSI protocol stack, either on the request of the user or as dictated by the Security Policy and management, in both their connection-oriented and connectionless forms.

It is worth noting that while confidentiality can (in theory) be guaranteed (that is, disclosure of confidential information can be prevented by using cryptographic techniques) it is impossible to *prevent* fraudulent modification of information in an untrusted network. The integrity function can only *detect* such modification and, optionally, make a serious effort to recover from it. Similarly, authentication function can only *verify* the identity of the other party and detect any changes, not guarantee that the other party remains the same. When an unrecoverable security violation is detected, the communication system can only disconnect and report the incident. The situation then reduces to ordinary denial of service, which again cannot be prevented in an untrustworthy network.

3.4 On the Formal Analysis of Secure Open Systems

The preceding analysis of security functions and mechanisms, together with the new concept of *security context*, gives us new insight to the relations between the security functions and mechanisms of open systems and enables us to split open systems security into chunks of manageable size.

With our security model, these informally stated ideas can be formalized and brought within the reach of theoretical study. Just like the well known properties of any "good" communications protocol (completeness, dead-lock freeness, etc.) a number of requirements for secure protocols, based on real security needs of the user, can be formulated and protocols fulfilling these requirements can be designed and implemented.

As defined in section 3.1, a security context between two communicating application instances is the union of the security variables of all the OSI layers of these two application instances. In this way, security can be split by the layers of OSI. We have also seen that security can be divided in the dimension of time and seen as a series of consecutive security contexts. Furthermore, it is possible to split security into ortho-

gonal functions (authentication, integrity and confidentiality) which can be treated separately. Here again we have to beware of the subtle interdependencies between these functions.

While the complete formalization of these ideas is beyond the scope of this study, we shall elaborate on them a bit further and present a semi-formal description of some aspects in order to clarify them and point out a direction for further study.

In the OSI security architecture no clear relationship between the connection-oriented and connectionless security functions and mechanisms was shown. Now we have discovered that the connection-oriented functions and mechanisms really are based on connectionless functions and mechanisms, which are the more fundamental building blocks of secure open systems.

In the example at the end of section 3.3 we could see that in order to set up secure connection-oriented communications connectionless functions first need to be applied to the messages exchanged during connection establishment. Only by applying the connectionless security functions and mechanisms to the first couple of messages exchanged can we gain confidence in the identity of the other party and exchange connection-specific secret information with him in a secure and integral manner. This in turn enables us to switch into a new security context specific to this connection and known only by the two communicating application instances.

This simple revelation opens up interesting new views. By developing a formal technique for describing and analyzing the security contexts and the transfers between them we could formally deduce and prove things about the security of open systems. In the area of formal protocol research there are a number of well established requirements for a "good" telecommunications protocol and a number of formal methods for describing and analyzing various protocols.

Among the commonly accepted basic properties of a good protocol are the following:

- A protocol has to be completely and unambiguously defined.

- A protocol must be free of dead-locks and live-locks.

- A protocol has to have an initial state.

- After error situations the protocol must return to a certain state (usually the initial state) within a finite period of time.

Surprisingly, no such list of properties has so far been published for security protocols. As a first approximation we could state some rather obvious requirements for a "good" security protocol:

- It has to have an initial security context.
- After error situations the protocol must return to a certain security context within a finite time.
- All the security contexts employed have to be "secure" in the sense that we can rely on the security functions provided in these contexts.
- The changes between the security contexts have to be secure.

After developing formalisms suited for defining and analyzing the security contexts and changes between them we could prove something about the security of an open system in the following way:

- Define the security contexts employed.
- Split up the contexts by the layers of OSI if necessary.
- Prove each context secure (layer by layer if necessary).
- Prove the changes between the contexts secure.

The development of a formalism for the analysis described above is obviously a rewarding task and should not prove too difficult for a mathematically oriented scientist. It is, however, beyond the scope of this work and proposed here as an area of further research. Rainer Rueppels paper [Rue90] appears to provide a good basis for developing a formalism for this purpose.

3.5 On the Placement of Security Functions

Originally the OSI reference model was based on the assumption that all communication is connection oriented and end-to-end from the transport layer up. Connectionless transport service was later included because it is more natural with the client-server model. Also relayed services, mainly electronic mail, are becoming increasingly important. As discussed earlier, the basic OSI model is not very well suited for this kind of use and leaves lots of the functionality of such services outside the OSI world and entirely to the application.

In a connection-oriented end-to-end service a transport connection is dedicated to serving one end-to-end instance of communication. For this reason, any security functions can be placed at the transport layer with end-to-end significance and sufficient granularity.

With a relayed service, such as MHS, the situation is in many ways analogous. In the OSI sense each hop is an instance of end-to-end communication. However, messages of several users are transferred across the same connections and the messages are raised to the level of the application process and stored at each intermediate node.

From the user's (or information system builder's) point of view, each hop corresponds to one sub-network in the previous case. Similarly, the routing function performed by the MHS (within the application), based on the information on the envelope of the message, is comparable to internet routing. Finally, part of the envelope (P2 in figure 2-2) and the message content is transferred truly end-to-end and corresponds to the TPDU (or NSDU) in the previous case.

Therefore, it is natural that functions which can be placed at the transport layer in true end-to-end communication shall be placed within the application in services such as MHS.

In connection-oriented end-to-end communications, security measures at layers 1 through 3 can be used to further enhance the security of the service but not alone to guarantee it. Some functions, such as traffic flow confidentiality, can only be reliably implemented at these layers. In MHS, security measures at layers 1 through 6 (or 7) can be used to enhance security but true end-to-end security and sufficient granularity can only be achieved in the application.

If data is encrypted at layer N the headers of the PDUs of all the layers below N are sent in cleartext and are susceptible to traffic analysis as well as manipulation. Network Addresses need to be in cleartext, at least in the routing nodes within the network. The routers always can perform traffic analysis and therefore should be reasonably trustworthy. However, Data Link or Physical Layer Encryption between two consecutive nodes or Traffic Padding at the Network or Data Link Layer can be used to protect against traffic analysis on the links between the routers.

There are a number of possible places for the above-mentioned security functions, especially Authentication, Integrity, and Confidentiality, in the information system. Here, a brief overview of the advantages and disadvantages of various placements is given. The structure of the application layer is explained e.g. in [X.200] or [IS9545].

3.5.1 Application Process

Placing these functions in the *Application Process* is a rather straight-forward solution. This approach, however, suffers from a number of shortcomings and should therefore be avoided. Among the reasons why this is not acceptable as a general solution are the following:

- This means having to define and implement the same security functions and mechanisms separately for each application leading into excessive work and needless duplication of functionality.

- This approach contradicts the principle that security should be an integral part of the communication service provided by OSI, functions and mechanism included in the application process being mainly beyond the scope of OSI.

3.5.2 Application and Presentation Layers

Security functions and mechanisms can be placed in the *Specific Application Service Elements* (*SASEs*) within the Application Layer of OSI. From the application process' point of view, these security functions are then a part of the communication service. However, should another SASE need the same functions and mechanisms, they will have to be rewritten for that SASE (and application).

Security functions and mechanisms can also be placed in the common *Application Service Elements* (*ASEs*), such as the *Association Control Service Element* (*ACSE*, defined in [IS8649, /A1, /A2, /A3, X.217] and [IS8650, /A1, /A4, X.227]). The service is now available to many SASEs and the same implementation can serve a wide range of applications (but not necessarily all of them). For example, ACSE currently provides the means to exchange authentication information coming from the SASE, still leaving much of the functionality to the SASE.

A natural solution to avoid duplicating the security functionality for each SASE and yet avoid the limitations of ACSE is to add another Common Application Service Element positioned between ACSE and the SASEs, namely the Secure Communication Service Element (SCSE as proposed in [NS89]).

Figure 3-5 illustrates the security architecture of an application, such as FTAM. The SASE relies on the security services offered by the SCSE. SCSE interacts with the *Security Management Information Base* (*SMIB*) and the X.500 Directory Service (X.509, for distributing certified public keys). SCSE in turn uses the authentication exchange mechanism of the ACSE.

The functions that naturally belong to the SCSE are *Authentication* (with the help of ACSE), *Integrity* and *Confidentiality* (especially Selective Field versions of the latter two).To complement the services that SCSE produces using its own functions and mechanisms, it also uses the presentation context management service of the Presentation Layer and the bulk integrity and confidentiality service provided by the *End-to-End Security Protocol* (*EESP*) [ISO90a] at the Transport Layer.

In [ZE90] it is proposed that the simple one-way authentication mechanism of the current FTAM protocol be extended to provide strong authentication. A working prototype of this scheme has been implemented at Swedish Telecom. While this approach has the advantage of complying with the standards as far as possible it cannot be expected to gain ground because of its application specific and ad-hoc nature.

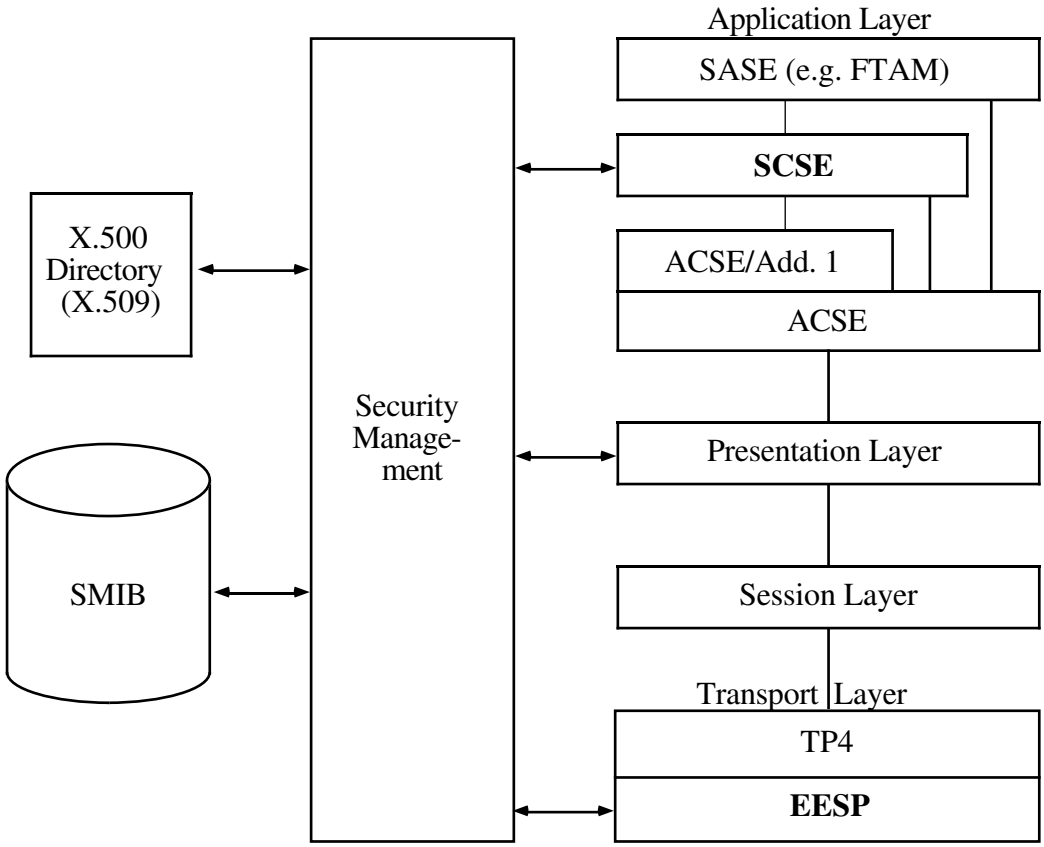

Figure 3-5: The proposed security architecture for an application such as FTAM.

In [NS89] the *non-repudiation* function is completely ignored (not even explicitly left outside the scope of the paper), probably because of its strong dependency on the application. However, the SCSE is also a natural place for a *generic non-repudiation function*. This function can provide non-repudiation of the sending of an *Application Protocol Data Unit* (*APDU*) without any regard to its semantics. This service can then be utilized by a variety of SASEs for various purposes. In this respect it is analogous to

the *synchronization function* provided by the Session Layer with no semantics associated with the synchronization points.

It is assumed in [NS89] that Selective Field Confidentiality should be provided by dividing a *Presentation Service Data Unit* (*PSDU* = APDU) into several *Presentation Protocol Data Units* (*PPDUs*), where each PPDU would be entirely encrypted or clear-text. The reason given for this is, that the current presentation layer does not support the use of more than one *Presentation Context* (Transfer Syntax) with one PPDU.

The approach proposed in [NS89] has several serious shortcomings, among which are the following:

- The segmentation and reassembly of the PSDU (APDU) at the Presentation Layer introduces a new complicated mechanism causing major changes and excessive over-head at the Presentation Layer.

- The need to change Presentation Contexts between each resulting PPDU adds further to this over-head.

In fact, this extra complexity appears to be completely unnecessary. A proposed simpler solution is the following:

- Enhance the ASN.1 data definition language [IS8824, IS8824/A1, X.208] (and the associated Abstract Syntax) to include qualifiers *Encrypted* and *Signed* applicable to any (structured) data type.

- Define Transfer Syntaxes capable of handling these new data types.

It is obvious that the macro facility of the ASN.1 definition language will undergo a total renovation in the near future for reasons discussed e.g. in [Ros90]. It is also probable, that ASN.1 and its *Basic Encoding Rules* (BER) [IS8825, IS8825/A1, X.209] will be extended to support cryptography instead leaving these things to be defined with macros. For reasons of both security and efficiency, BER should be redesigned in the near future. E.g. [HD89] proposes a transfer syntax which is claimed to be one to two orders of magnitude faster than BER. In the current version of the X.509 Authentication Framework [X.509], the following ASN.1 macro definitions for these functions already exist (see figure 3-6 below).

With the change in the roles of the Application and Presentation layers to be proposed here, the Application Layer would be responsible for performing the encryption and signature functions while it would be the task of the Presentation Layer to negotiate and manage the appropriate Presentation Contexts for this Security Context.

```
ENCRYPTED MACRO ::=
BEGIN
TYPE NOTATION     ::= type (ToBeEnciphered)
VALUE NOTATION  ::= value (VALUE BIT STRING)
END

SIGNED MACRO ::=
BEGIN
TYPE NOTATION     ::= type (ToBeSigned)
VALUE NOTATION  ::= value (VALUE
        SEQUENCE{
                ToBeSigned,
                AlgorithmIdentifier,
                -- of the algorithm usedto compute the signature
        ENCRYPTED OCTET STRING
        -- where the octet string is the result
        -- of the hashing of the value of 'ToBeSigned' --}
END -- of SIGNED. )

SIGNATURE MACRO ::=
BEGIN
TYPE NOTATION     ::= type (OfSignature)
VALUE NOTATION  ::= value (VALUE
        SEQUENCE{
                AlgorithmIdentifier,
                -- of the algorithm usedto compute the signature
        ENCRYPTED OCTET STRING
        -- where the octet string is a function (e.g. a compressed or hashed
        -- version) of the value 'OfSignature', which may include the
        -- identifier of the algorithm used to compute the signature --}
END -- of SIGNATURE. )
```

Figure 3-6: ASN.1 macros ENCRYPTED, SIGNED and SIGNATURE [X.509].

In practise, data to be signed would first be transformed into a Canonical Form (that is a fixed Transfer Syntax applicable to this Security Context, such as the ASN.1 BER) and then encrypted (with the key applicable to this Security Context). The resulting binary data would then be handed over to the Presentation Layer as an octet string. With our new arrangement the Presentation Layer is no longer interested in the structure or semantics of the PSDU, which is in harmony with the Layer Independence Principle of the OSI reference model.

As for signing data, a Digital Signature Scheme applicable to the current Security Context is employed in way similar to that described above for encryption. Here it is also possible to describe the Digital Signature mechanism in terms of ASN.1 and

employing a PKC without ever using the key word *Signed*. However, the *Signed* qualifier still needs to be included in the ASN.1 definition language in order to respect the orthogonality of Signature and Public Key Encryption and facilitate possible future Digital Signature Schemes.

According to the OSI reference model [X.200], the Presentation Layer provides session services and the following facilities: *transformation of syntax* and *selection of syntax*. There are three syntactic versions of the data: the syntax used by the originating application entity, the syntax used by the receiving application entity (the local syntaxes) and the syntax used between the presentation entities (the transfer syntax). The presentation layer contains functions necessary to transform between these syntaxes and each presentation entity must know the syntax used by its application entity.

In the OSI reference model the principles used to determine the seven layers of OSI are given. Among these principles are the following [X.200]:

P2: create a boundary at a point where the description of services can be small and the number of interactions across the boundary are minimized;

P5: select boundaries at a point which past experience has demonstrated to be successful; and

P9: allow changes of functions or protocols to be made within a layer without affecting other layers.

However, the current boundary between the application layer and the presentation layer seems to violate all of these three principles. With the current division data structures should be passed between the Application and Presentation Layers in the local syntax. Consequently, Presentation Layer would need to know about the structure of Application PDUs to be able to convert between the local and transfer syntaxes. This creates an unnecessary dependency between the two layers and causes any changes made in the application layer PDUs to affect the presentation layer. Real implementation work of open systems, such as the work with C-VOPS [Kar87], has shown that in practice syntax conversions have to be performed at the Application Layer.

The division of responsibilities between the Application and Presentation Layers should be revised. The proposed remedy is to let the Application Layer be responsible for the conversions between the local and transfer syntaxes and only leave the negotiation and management of Presentation Contexts to the Presentation Layer.

Among the benefits of this new arrangement would the following:

- In the new arrangement the Service Data Units transferred between any two OSI layers would be included in the Data PDUs of the lower layer as such, without any modifications, except possible encryption. No layer would need to know anything about the internal structure of its SDUs. That is, the Presentation Layer would not need to know about the syntax of the Application PDUs.

- Encoding and decoding of APDUs could be done more efficiently in the Application Layer, without the need to pass control information related to this process between the two layers. This would also make the description of services at the boundary between the application layer and the presentation layer simpler (see principle P2 above).

For the reasons stated above, practically all the upper layer security functions and mechanisms, including data encryption, are actually placed in the Application Layer in this study. Only the definition and management of transfer syntaxes and presentation contexts is left to the Presentation Layer.

Even though this may seem to be a fundamental change to the universally accepted OSI reference model it still can be done without serious consequences. OSI should only be concerned with the external protocol properties of telecommunications systems, not with their internals. The modification proposed here would not change the external behavior of an open systems, only make it internally cleaner.

The architecture proposed in [NS89] has since then been developed further by a group of Korean experts who have also implemented their SCSE proposal to run on a Sun 3 workstation and ISODE 5.0 [ISO90f]. The paper discusses the functions of authentication, integrity and confidentiality. Plans for extending the design to cover non-repudiation and access control as well as for improving the efficiency of the test implementation were also mentioned in this paper.

3.5.3 The Lower Layers

Most security functions and mechanisms can be placed at the Transport or Network Layers of OSI, so that they are available for various protocol stacks and applications. The main difference between placing these functions and mechanisms into one or the other of these two layers is that a transport connection is always serving exactly one instance of application level communication between two systems whereas a network connection can be multiplexed among several transport connections.

Also, in connectionless-mode communication, datagrams at the Network layer are host-to-host, whereas datagrams at the Transport Layer are process-to-process (that is they serve one instance of process-to-process communications, or one application association). Even though functions and mechanisms placed in the upper part of the Network Layer can provide end-to-end security functions (in the host-to-host sense) they cannot, in general, provide sufficient granularity, that is they cannot be focused on individual instances of user-to-user communication. For these reasons, the Transport Layer usually is a better place for these functions and mechanisms than the Network layer.

Unfortunately the question of placing certain functions at the Transport or Network layer often becomes a political issue. Transport layer being end-to-end, functions placed in the Transport layer or above it are entirely in the domain of terminal equipment and outside the control of teleoperators. For this reason there is usually a strong tendency to place functions and mechanisms in the Network (rather than Transport) layer among the teleoperators participating in international standardization. It is anticipated that the current liberalization trend in the area of telecommunications will, in the long run, solve these political problems.

4 A Secure General Purpose OSI Protocol Stack

In this chapter we shall place the security functions and mechanisms that we have discussed in the preceding chapter and in the appendix into the layers of OSI and find the appropriate mechanisms for realizing them. This placement is based on the criteria and considerations presented in chapter 3 and the appendix. To keep the work on a sufficiently general level, we do not not employ actual low-level mechanisms, such as cryptosystems, but keep the discussion on the level of protocol logic and abstract cryptographic mechanisms. Low-level mechanisms are dealt with in chapters 6 and 7, where real cryptosystems, real software environments, and concrete modifications to the existing protocols and their PDUs are suggested.

The functions and mechanisms are placed into a general purpose OSI protocol stack which can be used for various kinds of applications. Both connection-oriented and connectionless communications are considered and the same protocol stack can also be used to enhance the security of relayed services such as MHS.

Our goal here is as general a solution as possible, applicable to a large group of services and not tied into existing cryptosystems or protocols. It is realized that individual cryptosystems and even protocols will evolve and the solutions presented here should outlive them.

In order to accommodate for a number of various applications, options for using or not using various security functions need to be provided. Depending on the requirements of the specific application and economical reasons we may, for example, want to encrypt all, some or none of the data being transferred.

One instance of using a service across a network is illustrated in figure 2-1. Layers 1 through 3 of the OSI system are shared among various users, services and service instances. It is therefore not feasible to implement security functions focusing on a single service instance in these layers. It is possible to embed security functions and mechanisms in the lower layers, but they will only be effective across one hop or host-to-host, not from application process to application process.

Some possible uses of security functions in layers 1 through 3 are the following:

- Use of physical layer encryption to provide for traffic flow confidentiality and encryption of all traffic across a critical communication link.

- Use of data link level encryption on an untrusted link (such as radio link) or authentication, confidentiality, and integrity functions between pairs of stations on a broadcast medium (such as Local Area Network or Packet Radio).

- Use of network level security functions to secure communications between two hosts on the same subnetwork or across an internet.

Although access control should primarily be left to the application, it is often useful to have multilevel access control in a distributed environment in order to stop dummy traffic and most attacks as early as possible.

- Data link level access control can be used in broadcast media, such as LANs, to limit access from untrusted (or to critical) hosts or networks. For example, all messages arriving from a gateway to a critical host can be ignored.

- Similarly, network level access control can be used to limit data flow to and from critical hosts.

However, these measures alone should never be considered adequate and they should only be used to limit the number of attacks ever reaching the upper layers of information systems. If most attacks can be dealt with at the lower layers of the host machines (or even by dedicated gateways and other telecommunications equipment), more resources are spared to deal with the more serious attacks. All of these techniques merely serve as supplementary measures and do not provide the granularity or level of service necessary for secure use of various services by various users across a large, untrusted network.

Because layers 4 through 7 are end-to-end and no multiplexing takes place above the transport layer, that is, each (N)-connection (n=4,7) serves exactly one application association (service instance), it seems natural that all security functions and mechanisms relating to a single service instance should be placed in these layers. Already in 1983 Voydock and Kent concluded that end-to-end security measures, accomplished by adding security mechanisms in the high-level protocols (those of layers 4 through 7) of OSI, are more appropriate in an OSI environment than link-oriented security measures [VK83].

This work gives the first concrete and complete suggestion as to how the security functions and mechanisms described in the OSI Security Architecture should be placed in the layer of OSI in such a way as to considerably limit the degree of freedom in their placement without affecting the level of security offered by the OSI system. A well-founded placement of these functions and mechanisms is assumed to be of great value in defining and building secure open systems in the near future.

4.1 Authentication

In the open systems environment there are many types of entities which may be identified, such as physical entities (e.g. real open systems), logical entities (e.g. OSI layer entities) and human entities. Identification associates an identity with an entity. The process of corroborating an identity is called authentication [IS10181-2].

4.1.1 Authentication and its Significance with Various Services

As indicated earlier in this thesis, reliable authentication of the other party of communication is probably the most urgent problem in today's integrating, world-wide networks. Authentication can be divided into two classes:

- Peer Entity Authentication, used with connection-oriented communications.

- Data Origin Authentication, used with connectionless and relayed communications.

Peer Entity Authentication only authenticates the entity participating in the authentication exchange, it cannot alone guarantee that the other party remains the same during subsequent communications over the established connection. In Peer Entity Authentication a secure exchange of a session key in connection with the authentication exchange is needed for continued authentication. This leads into the need to not only sign but also encrypt at least parts of the authentication messages.

The secure exchange of a fresh pairwise session key completes the establishment of a Security Context, effective on this connection, where only the two mutually authenticated parties are in possession of this session key which can be used for continued authentication as well as purposes of data confidentiality and integrity.

When using a service across a network, we have to be able to reliably authenticate the other party. This means the user being able to authenticate the service and server he is using and the server being able to authenticate the user and his end system.

Ultimately, data security means giving only the legitimate *users* access to the data and services they are authorized to access. Therefore, it is essential that the user, not his system, be in possession of the keys required for authentication. However, it is often necessary to authenticate the end system as well, to ensure that:

- A legitimate user is not trying to access the service from a disallowed host (e.g. at gun-point from the enemy's system or voluntarily from a system whose security cannot be guaranteed).

- An illegitimate user is not using the service having first somehow obtained the keys necessary for authentication.

So, we need to be able to authenticate the service, its user, or both, depending on the specific service and its security requirements.

4.1.2 Placement of Authentication in the OSI model

The application layer is the appropriate place for authentication because only the application layer knows about various services, service instances etc. Therefore, the service and its user can only be reliably, and with sufficient granularity, authenticated at this layer.

While the authentication function is provided by the Application Layer, parts of the functionality need to reside above the OSI model. When authenticating the user, it is essential that he need not give his keys to the system, even though the application layer can provide the mechanism for user authentication. For example, a careful user would never feed in his secret authentication and signature keys to an untrusted sales terminal. However, it is necessary that the terminal be able to authenticate the user to the banking system and have him approve the money transfer with his digital signature.

The Authentication Function logically belongs to the proposed new common SASE called SCSE (Secure Communications Service Element, see [NS89]). SCSE can, in turn, use the authentication exchange mechanism defined in the security addendum of the Association Control Service Element (ACSE) [IS8649/A1].

As an intermediate solution, it is possible to offer authentication function at the transport layer. The proposed secure "network" (or in fact transport) protocol [ISO90a,b] provides us with this option and the secret information used in the authentication can still be supplied from the upper layers. While this is not a very neat solution, it gives us most of the benefits of authentication at the application layer and can be used in connection with virtually any service without us having to wait for the upper layer protocol standards to reach maturity.

4.1.3 Authentication Mechanisms

The two-way authentication protocol described in the appendix can be realized by using the authentication information exchange facility of the ACSE Security Addendum

[IS8649/A1]. Because ACSE only provides us with the means for a two-way authentication exchange, it is vital that we have a reliable two-way authentication protocol to implement with ACSE.

4.2 Integrity

4.2.1 Integrity and its Significance with Various Services

Together with Authentication, Data Integrity is usually considered the most important security function. In fact, functions such as Authentication and Non-repudiation are not of much value if not combined with Integrity. Being able to know, or even prove, the identity of the other party is useless if we cannot be certain that the information exchanged has not been tampered with. Similarly, knowing that the information is intact does not mean much if we cannot be sure whom it came from.

4.2.2 Placement of Integrity in the OSI Model

According to the OSI reference model, the Transport Service provides transparent transfer of data between session entities relieving them from any concern about how the reliable and cost-effective transfer of data is achieved. The functions invoked at the transport layer to provide the requested quality of service depend on the quality of the underlying network service. The selected quality of service is maintained throughout the lifetime of the transport connection and the session entity is notified of any failure to maintain the required quality of service on the transport connection [X.200].

Among the end-to-end functions of Transport Layer already are the following: sequence control, error detection and monitoring of the quality of service, and error recovery. Because Transport Layer is responsible for end-to-end error detection and recovery, it is natural to extend the error detection (and correction) functions and mechanisms of the transport layer to detect (and as much as possible recover from) malicious attacks against integrity as well as transmission errors. Activation of these mechanisms can be made easily, as an extension to the selection of functions describe above already taking place when opening a transport connection.

Being the lowest end-to-end layer, transport layer is the lowest possible layer for an end-to-end integrity function and mechanism. This mechanism could also be placed on top of the network layer. However, this is not an acceptable solution because network connections can be multiplexed (and in fact network service is increasingly often connectionless). Only in layers 4 through 7 does each connection serve just one instance of communications thereby offering sufficient granularity for the provision of integrity and confidentiality.

When using a sophisticated transport protocol, such as the ISO Class 4 Transport Protocol (ISO TP4) or the Department of Defense Transmission Control Protocol (DoD TCP) [RFC81b], reliable end-to-end protection is provided against accidental modifications of the data stream between the two transport entities. In this case, strong end-to-end integrity is easily achieved by placing an encryption mechanisms below the transport protocol as proposed in the appendix.

It is, therefore, suggested that the Integrity Function be placed at the Transport Layer. This applies to both Connection-oriented and Connectionless Transport Service when the integrity of the whole TPDU is to be ensured.

With some applications, Selective Field Integrity is desired. This function needs to be provided at the Application Layer because no lower layer is capable of dealing with various fields of the APDU. It is suggested, that Selective Field Integrity be placed in the Secure Communications Service Element (SCSE) of the Application Layer. This arrangement requires some support from the Presentation Layer. New data type "Signed" has to be included in the ASN.1 specification language and transfer syntaxes capable of handing this new data type need to be deviced.

It is also possible to cope with just one new type "Encrypted" and define signature mechanisms based on encryption by means of ASN.1. However, this is not a neat solution because digital signatures should be considered separately from PKCs, even though most current signature schemes are based on the use of PKCs.

The use of transport layer gateways complicates the provision of end-to-end security services at the transport layer, because then the transport connection is not truly end-to-end. When this is the case, two approaches are possible:

- Use trusted gateways within your own organization. This approach contradicts with our original goal of minimizing the use of trusted parties.

- Use the selective field integrity function of the application layer.

4.2.3 Integrity Mechanisms

Data Integrity is closely coupled with Confidentiality and can be provided with an Integrity Check Value calculated by using symmetric or asymmetric encryption mechanisms. Usually data encryption, combined with redundancy in the cleartext data, is sufficient to provide Integrity as a by-product of Confidentiality.

With connection-oriented services, it is suggested that non-selective "bulk-integrity" be provided by means of a secure transport protocol of the type SP4, EESP, or their derivatives.

SP4 is an encapsulating protocol placed at the bottom of the Transport Layer and encrypting all TPDUs of the actual Transport Protocol. EESP is a derivative of SP4, fixing some of the problems found in the preliminary SP4 specification. However, this proposal too has some faults which need to be corrected. SP3 is the counterpart of SP4 placed at uppermost part of the network layer. An introduction to SP3 and SP4 is found in [Nel88] and to SP4 in [BDH88]. For more precise definitions of the protocols see [SDN89a, SDN89b, ISO90b]. In [Hou89] a protocol based on SP4 is proposed to enhance the Xerox XNS (Xerox Network Systems) protocol suite with the functions of authentication, confidentiality and integrity. SP4-like functionality is also proposed in [Bir85] to be used with remote procedure calls.

The protocols mentioned above always do offer Message Integrity. That is, they protect each individual TPDU from modifications. With connection-oriented communications, Message Integrity alone is not sufficient but Sequence Integrity is required. This means that not only each TPDU but also the entire TPDU sequence is intact. That is, no TPDU has been deleted, added, repeated, or misplaced without this being detected.

Because the entire TPDU is encrypted, it is sufficient for Sequence Integrity that the TPDUs carry sequence numbers. This is the case with TP4 but, alas, not with TP0. If TP0 is used the Security Protocol has to be augmented with sequence numbers. A proposal to that direction is found in [CGV90]. Also, the current EESP proposal does not detect TPDUs omitted from the end of a transport connection if TP0 is used. TP4 again has sufficient functionality (a graceful disconnect procedure with negotiation of the last TPDU sequence number) to detect such attempts.

It is suggested, that a pairwise session key be generated for each connection, securely exchanged in connection with the Authentication Exchange, used with the Transport Layer Security Protocol, and disposed of at the end of the connection. This pairwise key forms an important part of the Security Context.

With connectionless services, only Message Integrity can be provided. Here setting up a Security Context including a pairwise symmetric key is impractical unless a predefined Default Context exists. It is, therefore, suggested that each message be signed with the secret key of the sender. This procedure offers not only Data Origin Authentication but also the stronger function of Non-repudiation of Origin.

4.3 Confidentiality

4.3.1 Confidentiality and its Significance with Various Services

Confidentiality is usually the first function that people associate with data security. However, it is generally not as important as Authentication or Integrity. Most of the data passed through a network is not very critical and the sheer bulk of it makes it difficult for the enemy to find the relevant pieces of information and make use of them. However, some of the data is always critical and should not be sent in cleartext. Most notably passwords and usernames must always be considered pieces of such critical information, as mentioned in section 1.2.

Confidentiality should be used to make gaining access to the data more expensive for an enemy than the value of the information revealed would be for him. This requires risk-assessment and security classification of data. It should again be noted that the Security and Integrity classes of the same information may be quite different. For example the Integrity Level of a public record is very high whereas its Security Level is nil.

However, confidentiality is vital for many applications, such as Electronic Data Interchange (EDI), Electronic Funds Transfer (EFT) etc., and it can often be provided at no extra cost, as a by-product of Integrity.

4.3.2 Placement of Confidentiality in the OSI Model

Confidentiality is closely coupled with Integrity and it logically belongs to the Transport Layer because it augments the Transport Service in a most natural way. Like Integrity, Confidentiality can be viewed as one more attribute of the quality of transport service. Also, since Confidentiality is usually produced by the same mechanism as integrity, it is natural to place these two functions at the same layer (and use the same mechanism to implement them both) in order to avoid unnecessary duplication of functionality.

In fact, based on the preceding discussion, Transport Layer is the only possible place for end-to-end bulk confidentiality for the following reasons:

- End-to-end Confidentiality cannot be placed below the transport layer because the lower layers are not necessarily end-to-end.

- Confidentiality should be placed below the transport protocol (such as ISO TP4) in order to provide end-to-end integrity at no extra cost.

Bulk confidentiality should, therefore, be placed at the transport layer in such a way that the actual TPDUs can be encrypted and that each transport connection can be protected independently of the other connections (that is, a different key can be used on each connection and not all connections need to be encrypted).

Just like with Selective Field Integrity, the function of Selective Field Confidentiality needs to be provided at the Application Layer because no lower layer is capable of dealing with various fields of the APDU. Selective Field Confidentiality is one of the functions provided by the SCSE of the Application Layer. Like Integrity, Confidentiality in the Application Layer requires some support from the Presentation Layer. What was said previously about the effect of transport layer gateways on integrity also applies to confidentiality at the transport layer.

4.3.3 Confidentiality Mechanisms

For Connection Confidentiality, the same Transport Layer Security Protocol as for Connection Integrity is used.

For Connectionless Confidentiality, Public Key Encryption with the public key of the recipient is used.

If a pre-defined (or otherwise agreed-on) common Security Context, including a pairwise symmetric key between the sender and the recipient, exists, it is possible to use symmetric encryption with this key for Connectionless Confidentiality. This common Security Context can also be set up by including the pairwise key, encrypted with the public key of the recipient, in the message. This procedure makes sense performance-wise if the message is very long.

4.4 Non-Repudiation

Non-Repudiation is a security function, which provides proof of the origin or delivery of data in order to protect the sender against false denial by the recipient that the data has been received, or to protect the recipient against false denial by the sender that the data has been sent. Non-repudiation implies the existence of a trusted third party, whose primary role is to arbitrate disputes resulting from Non-Repudiation [ISO90e].

4.4.1 Non-Repudiation and its Significance with Various Services

Data Communication is gradually replacing the transfer of paper documents within and between organizations. If Electronic Documents and direct transactions between information systems are ever to replace (or even seriously compete with) paper in inter-

corporate business, it is essential that there exist means to make commitments electronically.

Non-repudiation is especially important with services such as MHS, where electronic mail needs to be signed, EDI, where inter-corporate offers, orders, bills etc. are to be handled electronically, and EMT, where (possibly large) sums of money are transferred.

Non-repudiation has associated with it a hoard of legal problems that need to be solved. Before electronic commitments can be used in the court of law, a lot of legislative work, which is outside the scope of engineering as well as that of this study, is needed.

In any case, we must first define sound methods for making and showing true electronic commitments before any concrete legislation in this area is possible. Then there is a chance that laws defining the properties of an acceptable digital signature can be passed and a procedure for keeping and updating an official list of currently acceptable signature mechanisms, as well as an official record of approved CAs, can be set up.

4.4.2 Placement of Non-Repudiation in the OSI Model

Non-repudiation is strongly application-dependent but a generic non-repudiation function can still be included in the SCSE of the Application Layer. This function can be used to sign any APDU with no regard to its structure or semantics. This generic service can, in turn, be used by various SASEs for various purposes.

4.4.3 Non-Repudiation Mechanisms

Non-repudiation is usually based on digital signatures which, in turn, are currently based on PKCs. A message signed with the secret key of the sender can be proved to have been sent by the holder of the secret key, because nobody else could have generated the signature. Giving away one's secret key can be considered giving out an open proxy, even though a procedure for cancelling compromised keys will normally have to be provided. Cryptosums based on symmetric encryption can be used to make the generation and checking of signatures more efficient.

It is proposed that a Digital Signature Mechanism be embedded in the SCSE in the application layer. In terms of low-level mechanisms the signature mechanism operates in the following way:

- Convert the APDU into a canonical form.

- Compute a cryptosum of the APDU.

- Encrypt the cryptosum with the secret key of the sender.

- Attach this signature to the original APDU.

In the receiving end the signature is checked in a similar fashion.

In case of non-repudiation of receipt, it deserves to be noted that the mechanism must be activated *before* the data is transmitted to the recipient. Otherwise the recipient may first examine the data and then decide whether or not he wishes to acknowledge it. This leads into one of the two solutions:

- First deliver a digital finger-print of the message and require a signed acknowledgement from the recipient ("intent to receive"). Then deliver the whole message.

- Use a trusted security server to handle the delivery and require a receipt before submitting the message.

Neither of these two approaches completely solves the problem, as explained in [ISO90e]. Also, since we do not wish to include any more trusted parties than absolutely necessary, we decide, that a receipt obtained voluntarily from the recipient shall be adequate for our purposes.

Notary services can be used to support non-repudiation. It is, for example, possible to allow a trusted notary server to authenticate the communicating parties and register and certify commitments made between them. This kind of service can be offered in connection with, or instead of, the mechanism described above. This study concentrates on realizing secure communications with a minimal number of trusted parties (that is, we trust in only our own CA). Therefore, trusted servers are not dealt with in any detail here.

4.5 Access Control

4.5.1 Access Control and its Significance with Various Services

Access Control is not a security function in the same sense as the four security functions discussed above. For example, we do not explicitly ask for access control to be employed with a connection when opening it. Rather, access control has to do with Authentication as well as Network and Applications Management.

4.5.2 Placement of Access Control in the OSI Model

Access Control can be employed at most levels of OSI to limit illegitimate access to (or from) networks and services. In current distributed systems Access Control is performed by means of e.g. the following mechanisms:

- At the Data Link Layer, with filters in bridges.
- At the Network Layer, in routers with limitations based on network addresses.
- At the Transport Layer, in routers based on TCP Socket Numbers (and OSI TSAP-numbers).
- Within the Application.

The ultimate responsibility for Access Control is in the Application and therefore mainly beyond the scope of OSI.

Access Control in the lower layers (Transport, Network and Data Link) can be used to effectively reduce the number of serious attacks. Management of virtual networks can be used to aid in this. Current routers have Access Control Lists etc. which can be maintained by means of standardized network management protocols. Frame Relay techniques will make it possible to define virtual networks at the Datalink Layer.

4.5.3 Access Control Mechanisms

Access Control should be based on reliable (strong) authentication of the user and an information base kept by the application on the privileges and restrictions of each user or group of users. It is important to separate Authentication from Access Control. The Access Control Scheme included in the current FTAM standard [IS8571-2] is an example of mixing (simple, application specific) authentication and access control with disastrous consequences. This example demonstrates the necessity to find more general solutions to the security problems of open systems instead of solving these problems on a per-application basis.

5 Management Issues

All security functions and services can only be provided within the context of a defined security policy for a specific application. Ultimately, it is left as a responsibility of systems management to enforce this security policy by activating the appropriate security functions required by the security policy. International standardization of systems management is increasingly focusing on security management, as demonstrated by e.g. [IS10164-7,8,9].

5.1 Activation and Negotiation of Security Functions

Security functions can be activated in either of the two ways:

- On user request.

- By system management as dictated by the security policy.

In the general-purpose secure OSI protocol stack drafted in chapter 4, security is an integral element of the quality of service provided by the communications system. The user can indicate the security functions he wishes to activate by the Quality of Service (QOS) parameter when opening the communications. Only formal changes are needed in the current service interfaces to accommodate for this facility.

Each OSI layer can provide the security services requested that it is capable of providing and pass a request for the rest on to the lower layers. In this manner, the application is not greatly affected by the fact that some security functions are moved from one layer to another. For example, Authentication or Integrity could be moved from the transport layer up to the application layer without any changes to the applications using these functions.

The communication service provider need not provide exactly the security services requested by the user. For example, if the user requires data integrity, the service provider can activate an encryption mechanism offering both integrity and confidentiality. Also the system may have a policy of, for example, not offering integrity without authentication, or vice versa. Should the user ask for integrity, he may end up getting authentication and confidentiality too. Also, systems management may activate security functions without any interference from the user when dictated by the applicable security policy.

When negotiating the use of security functions and mechanisms between the communicating parties, a complicated negotiating procedure should be avoided. It is enough if

we let the originating party choose the set of functions it is willing to use. The responding party may either accept this offer, add its own requirements, or refuse the connection. Similarly the originating party can refuse the connection should the responding party require the use of additional security functions and mechanisms not supported by this end system.

The negotiation mechanism suggested above appears to be a reasonable compromise between simplicity and functionality. It can be easily implemented, it only uses a two-way handshake, and nevertheless it provides adequate functionality for most purposes.

5.2 Distribution and Management of Keys

Keys are divided into two categories:

- Master Keys, used for authentication, key exchange, and key management. They can also be used for encrypting short datagrams in connectionless communications. These keys are rather permanent in nature, they should be stored with utmost care and never be used for the encryption of bulk traffic.

- Session Keys, used for the encryption of bulk traffic. These keys are very perishable and should not be used for extended periods of time. They should normally be pairwise.

A third group of keys are those used for encrypting stored data. These keys have nothing to do with data transfer and their generation and secure storage is beyond the scope of this study.

In most security schemes, keys are referred to by key-IDs. The idea is to separate the use of keys from their distribution. Keys are stored in a database, referenced by the key-IDs, and used as needed.

However, this approach has a severe shortcoming. When using a key stored in the database we cannot have any knowledge of its freshness. In fact, this practise encourages sloppy usage of session keys. It is highly likely that in many cases a small collection of keys will be used for extended periods of time leading into repeated use of the same keys. Also, stored keys can leak before or after their use making it possible to decrypt recorded traffic afterwards.

Generating symmetric encryption keys is almost trivial and session keys usually are symmetric. Even strong public key pairs can nowadays rather easily be generated by using a cheap personal computer. It is therefore evident that labelling and storing session keys does more harm than good and this practise should be abandoned.

Instead, fresh session keys should be generated when needed and securely exchanged in connection with authentication by using the permanent master keys. Should the need arise to change session keys in the middle of a long session, this can easily be done by repeating the authentication procedure and its associated key exchange.

5.2.1 Management of Master Keys

The minimal procedures for the management of master keys are the following:

- User A registers to his CA exchanging Public Keys with it. The user can generate his own key pair himself, so that he need not trust anybody else with his secret key. The CA gives A a Signed Certificate of A's Public Key $K_{P,A}$: $S_{CA}\{CA, A, K_{P,A}, t_e\}$, where t_e is the expiration time of the certificate. Also the CA's Public Key $K_{P,CA}$ is handed over to A in such a way as to sustain its integrity.

- The CAs form a global hierarchy logically structured as a tree. Each CA in this tree certifies its father and children.

It is important to notice that the hierarchy of CAs need not have anything to do with the hierarchy of Directory Server Agents (DSAs) distributing the certificates. For example, the CAs may be administrative bodies while the DSAs may be administered by teleoperators and private companies.

To facilitate the possibility of using different key pairs for different purposes (e.g. one key pair for confidentiality, another for non-repudiation) we should include the intended area of use of the public key in each certificate. Because the PKCs used may (and will) change with time, we must also include an identifier of the algorithm for which the public key is intended. This is most easily done by letting the key value in the certificate be a structured data type with all essential information (such as algorithm identifier and scope of use) included. Considering the fast progress in the area of cryptology, we must be prepared for completely new types of keys, which may be, for example, large sets of data items. This should be borne in mind when certificates are designed.

In the CCITT recommendation X.509 a *Certification Path* is formed between the CAs of two parties served by different CAs and wishing to communicate in a secure manner.

Let us assume that A trusts CA_A and wishes to communicate with B belonging to the domain of CA_B. A now possesses an integral copy of K_{P,CA_A} (the public key of CA_A). However, CA_A cannot directly certify B because B belongs to a different administrative domain, namely that of CA_B. We now need to find a Certification Path between CA_A and CA_B, that is a chain of CA_i, such that: A => CA_A -> CA_1 -> CA_2 -> ... -> CA_n ->

CA_B -> B (A trusts CA_A, which certifies CA_1, which certifies CA_2, which certifies...which certifies CA_n,which certifies CA_B, which certifies B). This unbroken chain of trust implies that A can trust in the integrity of B's Public Key K_{PB} included in the certificate $S_{CAB}\{CA_A, A, K_{P,B}, t_e\}$ issued by CA_B.

This means that A has to acquire from somewhere (not necessarily from the directory) all the certificates necessary for forming this Certification Path (that is one Certificate for each arrow "->" in the formula above). These Certificates may be obtained from anywhere, not necessarily from different DSAs.

In a tree-like structure of CAs the shortest Certification Path is never overly long. It can be shortened further by having CAs with lots of traffic between their domains directly cross-certify one-another. It is not obvious how a reasonably short and reliable path is found. For example X.509 does not recommend any method for this.

The three basic approaches are the following:

- The originating party can find a certification path and pass it to the recipient along with the message.

- The responding party can find a certification path.

- The originator can provide part of the certification path and the responder complete the path.

These three modes of operation are not mutually exclusive. Obviously, the amount of trust that a user can place in a certification path cannot be greater than the amount of trust he is willing to place in the weakest link of the chain. If the responder does not trust in (parts of) the path provided by the originator and wants to find a safer certification path this option should be left open for him as proposed in [MWR89].

A simple solution for finding a path is to require that every communicating party if in possession of the (bi-directional) certification path between itself and the world root. The sender passes the half-path from the world root to itself along with the message. The recipient combines this information with the half path from itself to to the world root to find the certification path via the closest common ancestor in the tree of CAs. Optimization, e.g. based on knowledge about the uppermost level of the closest common ancestor, is possible but not necessary.

The normal way of distributing certificates is via the Directory by according to the CCITT recommendation X.509. However, certificates can be distributed through any channel and in the scheme described above each party only has to know the rather

stable certification path between itself and the world root making the need of directory queries in forming certification path rather infrequent.

5.2.2 Management of Session Keys

Ways of exchanging fresh keys in connection with authentication and using them exclusively for this one session should be studied. Here, no need to identify session keys exists, as long as they are mutually agreed upon. The session key never needs to be explicitly referenced, the key belonging to the current security context always being used. After the session the key is disposed of.

To guarantee the freshness of keys, at least three approaches are possible:

- Store hash values of old keys and refuse synonym keys.

- Use two xored half-keys, one chosen by each party, to form a session key.

- Use a trusted server (key distribution center) to generate fresh keys.

Old keys cannot be stored as such because they must not leak. Therefore, a one-way hash function needs to be applied on them before they can be stored. A proposed new key is first hashed and then checked against the hash values of used keys. If the values match, the new key is rejected, otherwise accepted as being fresh.

One way of achieving freshness is generating pairwise session keys by bit-wise exclusive or (xor) operation between two half-keys. In this scheme the communicating parties A and B choose their own half-keys K_A and K_B, respectively. The pairwise session key K_{AB} is formed as follows: $K_{AB} = K_A$ XOR K_B. In this scheme a protocol for securely exchanging the half-keys without A knowing K_B before it has sent K_A (and without B knowing K_A before it has sent K_B) is needed. The most straightforward way of achieving this is with the help of a trusted security server involved in the process.

Also, a trusted security server can be used to generate and securely distribute fresh, strong, pairwise session keys. However, this is not a desirable solution since we want to minimize the use of trusted functionality.

In practise, the freshness of randomly generated keys can usually be trusted without any extra precautions. Well designed authentication protocols suffice to prevent the replay of authentication sequences (and the malicious reintroduction of old, compromised session keys). It is, therefore, adequate that the originating party generates a random session key when needed and securely conveys it to the responding party.

5.2.3 On the Generation of Public Key Pairs

The generation of strong Public Key Pairs (e.g. for RSA) is a fairly simple task that can be performed, for example, on a microcomputer. However, for the non-technical user it is necessary that he be able to purchase Public Key Pairs from a trusted supplier. This supplier can be the CA the user registers to.

When the keys are generated by someone else than the user, the following procedure should be followed:

- A trusted piece of software and hardware is used for key generation. For example a dedicated microcomputer with the proper software. The software should first be inspected as source code, then compiled. The system used should be sealed after inspection.

- The system should work in such a way that when a key pair is generated the secret key is only output on a diskette (or other medium) belonging to the user. After this, the memory of the system is erased and the service provider cannot obtain a copy of the secret key.

It should be noted, that with zero-knowledge techniques the identification information cannot be generated by the user but it always needs to be generated by the CA. Compares with public key pairs generated by the user, this causes the extra problem of securely transferring the secret authentication information from the CA to the user and requires us to trust the CA to keep this authentication information secret (or dispose of it) and not misuse it.

In fact we always have to trust the CA in any case. If we want to use PKCs for authentication and encryption we have to trust in the certificates issued by the CA. Even if a crooked CA does not know the secret key generated by us, it can always generate a key pair of its own and issue a false certificate for the public part of it.

Similarly, we always have to distribute some general, public authentication information to all users. If we are using PKCs this is the public key of the CA. In case of zero-knowledge schemes it is the public verification information.

So, in fact zero-knowledge schemes don't introduce any new problems, except that of securely transferring the secret key to each user. This problem is more than made up for by the ease of key management, when no user-dependent public authentication information needs to be distributed but the same public information is used to authenticate any user. We can therefore expect the use of zero-knowledge schemes in authentication to rapidly grow in the near future.

6 Implementational Issues

The goal of this work is to come up with solutions for open systems security that are implementable. This chapter deals with the implementational issues associated with the solutions proposed here and seeks to demonstrate that it actually is feasible to implement a secure open system with a reasonable amount of work, based on the existing OSI implementations.

In this chapter the effect of various hardware and software solutions on the over-all security of an open system are analyzed. Also some existing cryptosystems are suggested to fill in the parts still missing after the preceding chapters.

6.1 Software Environments

Because the goal of this work is solutions that can be implemented with reasonable modifications to the existing open systems implementations, it is necessary to consider the existing software environments and their use in implementing the solutions proposed here.

6.1.1 Some Sample Environments

In the implementation work going on at Telecom Finland, two software environments are being used. These are the C-VOPS protocol development and run-time environment developed at the Technical Research Centre of Finland [Kar86, Kar87] and the internationally well-established ISODE software package, whose development was started by Marshall Rose [Ros90, PSI90]. A brief overview of these two environments is given here.

C-VOPS

Development of software tools for implementing open systems was started at the Telecommunications laboratory of the Technical Research Centre of Finland in 1983. This work led to the birth of the Virtual Operating System (VOPS) originally written in Pascal and running on a VAX/VMS minicomputer.

The main goals of VOPS were to develop an integrated environment for developing, testing, and running layered protocols that would run on virtually any computer and host operating system. Protocols were modelled as Extended Finite State Automata (EFSAs) which were then converted into state transition tables accepted by VOPS. The control logic was implemented as a data structure rather than in program code. This made the protocols much easier to modify and debug. The EFSA would contain the

sequential control logic (see fig. 3-1) of the protocol whereas the coding and decoding functions were written as separate subroutines and linked together with the system.

Each protocol entity instance runs as a virtual process of VOPS whereas the whole VOPS system is just one process from the host operating system's point of view, as illustrated in figure 6-1. In this way, the protocol entities only rely on the services offered by VOPS, such as scheduling, dynamic memory allocation, message passing, and timer services, and are completely independent of the host operating system.

The VOPS environment itself being written in a high-level language, porting it from one system to another is fairly straight-forward. All the host operating system dependent parts of VOPS were collected to a couple of modules. When these modules were rewritten, the whole system could be compiled for a new machine and all the existing protocol implementations could be used without any modifications.

The portability of VOPS was demonstrated in practise by using the same software in a number of systems, such as VAX/VMS, intel 8086/iRMX-86 and IBM PC/PC-DOS, with a very reasonable effort in porting the system. Also the same protocol implementations were used as parts of various protocol stacks, demonstrating the modularity of the system. For example, the first prototype transport layer gateway (just like the gateways described in [Ros90] and [LT90]) relaying between TP4/LAN and TP0/PDN was designed in 1983 [KLP83] and built in 1985 at the Technical Research Centre of Finland.

With VOPS the CCITT Teletex document transfer system, which at the time was the only standardized seven-layer open system, was implemented in 1983 through 1985. The implementation was validated by Telecom Finland in 1985 and turned into a commercial product (the first available Teletex implementation for VAX/VMS) by Scandinavian Softline Technologies Inc. This implementation was then used by a number of large Finnish corporations as the "transport system" in the first national EDI experiments.

In early 1986 the development of the second generation VOPS was started. C-VOPS (for C-language VOPS) was written in the C programming language, which is more portable, more modular, and better suited for systems programming than Pascal. At the same time a block-structured and more powerful C-like protocol description language was developed for specifying the EFSAs. Also an ASN.1 compiler and interpreter were developed to be used with C-VOPS [Koi87, Nok90].

C-VOPS was later adopted by several large Finnish organizations as their preferred protocol development environment. Telenokia Inc. chose C-VOPS as the basis for their GSM system development after comparing it with the other available environments, including ISODE and Retix.

There currently exists a large library of C-VOPS protocol implementations, including the most complete FTAM protocol stack available today, a complete MAP 2.1 implementation (completed in 1987), and the X.500 Directory (including X.509). For an overview of the current status of the C-VOPS protocol library see [AKS90].

The strengths of C-VOPS still are its portability, modularity, large number of existing protocol implementations, integrated testing facilities, and available licensing for commercial use. A current description of the C-VOPS tool and its features can be found in [HKK90]. The basic principle of C-VOPS is presented in figure 6-1.

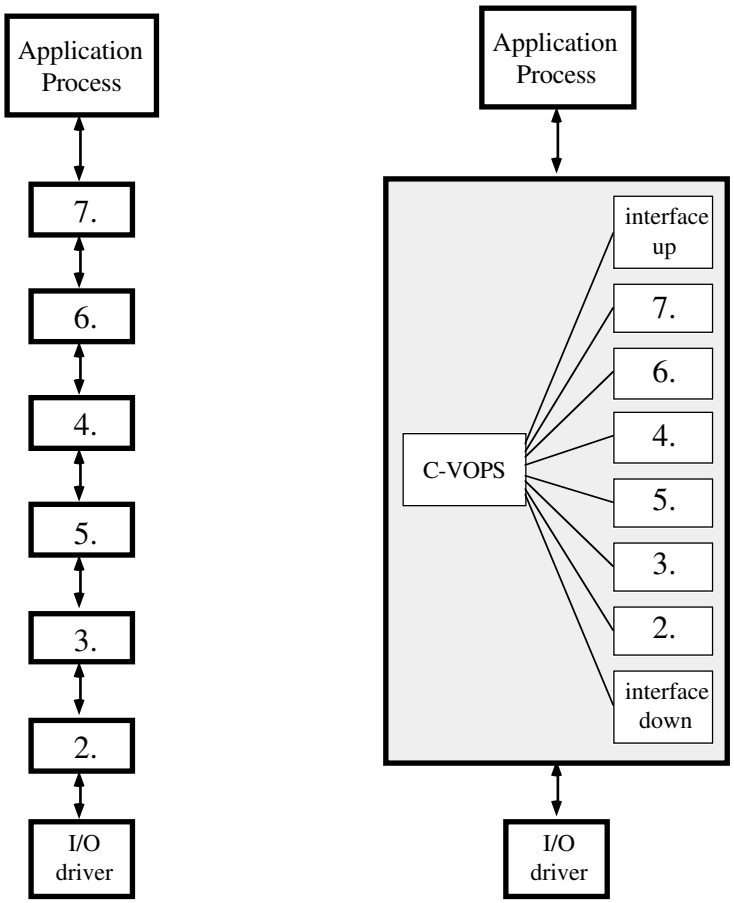

Figure 6-1: A conventional vs. C-VOPS -based open systems implementation.

The next generation of VOPS is already under development since 1988. The Distributed VOPS (D-VOPS) has all the good features of C-VOPS but it is also designed to be distributed to a number of processors. Compilers for the SDL [Z.100] and LOTOS [IS8807] protocol specification languages are also being developed for D-VOPS (see also [IS10167]). D-VOPS is described in detail in [KM91].

I believe, that the cheap and powerful microprocessors available today and the use of parallel processing, together with automatic and semi-automatic tools for converting high-level formal protocol descriptions into executable programs, will provide us with the keys to implement affordable and efficient open systems that can take full advantage of the transmission capacity of tomorrow's high-performance data networks. Sophisticated software tools, such as C-VOPS and D-VOPS, are of great value in this work.

ISODE

ISODE is in many ways fundamentally different from C-VOPS. Rather than a development environment, it is a large, well-structured collection of program modules with the appropriate *make* files etc. included. ISODE relies heavily on the system calls of the SunOS operating system, being initially developed on a Sun 3 workstation, and is not easily ported to other versions of Unix, let alone other operating systems. While C-VOPS tries to minimize the number of real processes (replacing them with virtual processes) ISODE *forks* a large number of processes in the best Unix tradition.

ISODE has rapidly become the most widely used OSI protocol development environment, both in the U.S. and in Europe. The current version 6.0 of ISODE includes many interesting things in addition to the normal FTAM and MHS, such as ASN.1 tools, FTAM/FTP gateway, OSI Directory Service, OSI Virtual Terminal, OSI Reliable Transfer, transport layer gateways between TP0, TP4 and TCP, and support for both X.25 and DoD IP networks [PSI90]. ISODE 6.0 is aligned with U.S. GOSIP [GOS89]. ISODE has in fact become the de facto reference against which other OSI implementations are frequently tested.

6.1.2 The Effects of Software Environment on Systems Security

The implications of the software environment on the overall security of the system must be stressed here. While in theory it is often significant in which layer a function is placed this may not be so with the actual working system.

It can, for example, be argued that peer entity authentication on the level of the FTAM SASE is more reliable than on the level of ACSE, which again is more reliable than authentication at the Transport Layer. However, this need not be the case. In an actual implementation at least layers 4 through 7 are usually running within the same software environment. For example, with ISODE they are running as Unix processes with the same owner, privileges etc., or with C-VOPS they are running as virtual tasks within the same process of the host operating system. Therefore, mechanisms in any of the layers 4 through 7 can be considered equally vulnerable (or resistant) to outside attacks.

Usually layers 4 through 7 of OSI are implemented within the same environment. Therefore, the degree of trust one can place in security measures located within these layers if practically independent of the layer. E.g. Confidentiality can equally well be placed in the Transport Layer or in the Application Layer as far as the size of the trusted software base is concerned.

6.2 On the Use of Hardware and Software

The price of encryption hardware is decreasing and its availability is improving. In the near future it will be feasible to encrypt all data with a symmetric cryptosystem at the full speed of current LANs (1-100 Mb/s). Encryption chips are still rather expensive and hard to get outside the U.S., mainly because of the policy of the National Security Agency (NSA) to limit the exportation and use of encryption devices abroad. In time this area will enjoy the benefits of the same kind of technological progress and commercial competition as we have seen in the area of microprocessors.

A study on VLSI implementations of PKCs is found in [BG89].

The use of signal processors can offer speeds comparable to hardware implementations with the flexibility of software implementations.

6.3 On the Role of Smart Cards

As mentioned earlier, it is desirable to authenticate not only the end system but also its user. This will make it possible for the user to use information services in a secure manner, make commitments etc. from any work station. Especially when using a work station other than that of his own, the user should not trust in it. At the very least he should never give his master keys (those used for authentication and key distribution) to an untrusted piece of equipment. Even giving a perishable session key to someone else is a security threat but at least this threat is limited in time and does not enable the enemy to make commitments on behalf of the legitimate user.

In fact, it can be argued that it is judicially essential that the functions of authentication and non-repudiation be raised on the level of the user, rather than left to his "agent" within the OSI world (that is his Application Entity). For example, it can be argued that it is not the work station of the President but only the President himself that should be allowed to declare a war.

In secure communications between application instances A and B, it is essential that at least some part of the security context SC_{AB} depend on the user in the following ways:

- Identification of the user is required when setting up the security context ("need to know") and activity from the part of the user is required for making commitments.

- The security context will be automatically terminated as soon as the user secedes from the transaction.

If the secret required for authentication and digital signature is possessed by the user, and can only be used by him, it is fairly simple to give this kind of authentication and non-repudiation a judicial standing by means of legislation. If, on the other hand, these operations were performed entirely by a piece of equipment (that is the user's end system), it would always be possible to dispute them later.

However, the cryptographic keys and functions involved in strong authentication, data encryption, and digital signature are too complicated to be memorized and performed mentally. This means that the user needs to store his keys somewhere and use a piece of trusted equipment to perform the cryptographic functions.

There clearly exists a need for a small, trusted personal piece of equipment, which is capable of securely storing the keys and performing the cryptographic functions needed and can only be activated by its legitimate owner. A smart card can be used to securely hold its owner's secret keys and include hardware mechanisms implementing the cryptosystems employed. For the activation of the smart card a biometric mechanism (based on e.g. recognition of retina patterns or finger prints) could be used but a Personal Identity Number (PIN code, comparable to that of the PIN codes of current dumb plastic cards) memorized by the user can be considered adequate.

The current smart cards are not suitable for our purposes for several reasons:

- They include functions that are not at all necessary but add to the complexity of the device, such as a "purse" for storing money.

- They lack an integrated keyboard, which is absolutely vital for safe authentication of the user.

To protect against misuse after theft or loss of a smart card, it must be activated with a PIN code. Without this PIN code the card cannot be activated and used. The security of a smart card depends on the secrecy of this PIN code, should it leak the smart card can be misused by anyone gaining possession of the card. Therefore, the PIN code should be memorized by the owner of the card and never be written down, or keyed into an untrusted piece of equipment.

What is an untrusted piece of equipment? Any public device, such as an automatic teller machine or vending machine, certainly has to be considered untrustworthy. One's private work station probably is relatively trustworthy, provided that it is sealed and kept behind locked doors. However, ultimately the smallest possible set of trusted equipment is the smart card itself. A minimum requirement for a truly secure smart card is that it has a keyboard of its own for keying in the secret PIN code.

On the other hand, some facilities commonly included in modern smart cards, such as a rechargeable purse for petty cash, will be completely unnecessary in the near future. With the integrating networks and real-time banking services emerging, authentication and non-repudiation (combined with integrity and confidentiality) will be sufficient functions for electronic purchases. Once the user is identified, any one of his accounts can be debited, in real-time, through the network. With small purchases, where the cost of real-time communications and processing would be unreasonable, vending machines and such can be operated off-line. Larger transactions can still be processed in real-time. The limit for checking the status of the card (e.g. reported stolen) and balance of the account can be made variable (from day to day and machine to machine) to further discourage the misuse of smart cards that have got into wrong hands.

The essential facilities a smart card must include are the following:

- Public key cryptosystem for encrypting and decrypting data and generating and checking digital signatures. It must be impossible to gain access to the secret key of the card by any means.

- Symmetric cryptosystem for bulk encryption and decryption (confidentiality and integrity of communications). It should be possible to load in pairwise session keys encrypted and signed with public key cryptosystems without the pairwise keys ever leaving the card in an unencrypted form.

- Memory to keep a log of the most recent transactions.

- Integrated keyboard to key in the PIN code.

- Preferably, a small display for displaying critical information.

The reasons for wanting all these facilities on the smart card are more or less obvious. Without an integrated keyboard PIN-codes would easily be compromised and the reliable authentication and non-repudiation functions of the smart card lost. A hostile machine could even swallow the smart card having first obtained the secret code for activating it. An asymmetric cryptosystem is essential for authentication, non-repudiation, and confidentiality.

A symmetric cryptosystem is needed on the card for efficient bulk encryption in order to achieve confidentiality, integrity, and continued authentication with connection-oriented services, as described in section 3.2.1. If this mechanism was not to reside on the card, the terminal would have to know the pairwise session key and could continue the transaction on behalf of the user even after the smart card has been extracted. Even though the terminal could not make commitments without the secret keys on the card, it could still impersonate as the user after the user has left. If the physical encryption mechanism, as well as the pairwise symmetric key, is on the smart card, the security context will automatically be terminated when when the user takes his card and leaves.

An integrated display is needed, because without a trusted system that includes a display the user cannot know for sure what he is signing. For example, a vending machine or an automatic teller machine could debit $1,000 from the user's account while the user thinks he is approving a transaction of $100. In fact, the machine could perform any number of transactions that the user is completely unaware of while his activated card is inserted in the machine. The way to prevent this is by letting the user's card display the essentials of a transaction before the user commits himself to it. For example, the card could display the message: "debit $1,000" after which the user could either approve or abort the transaction by pressing a key on the card's keyboard.

The log memory is needed in order to settle any possible disputes later. Its existence alone considerably reduces the chance of a dispute. In the log memory the card can store an unforgeable receipt, signed by the other party, of each transaction made with the card. The user can periodically download the contents of the log e.g. into his personal work station or a commercial (or official) trusted machine, thereby releasing the memory space for reuse.

The chip area, and therefore also the storage and processing capacity, of current smart cards is rather limited, as pointed out e.g. in [FP90]. Also display tehcnologies are not yet sufficiently advanced for building small, cheap, and flexible displays. It is clear that all of the facilities discussed above will not fit on a smart card for at least several years. However, it can be argued that a smart card without these facilities cannot be used with untrusted terminals and it can give the user a false impression of security thereby being even less safe than the magnetic stripe card it superseded.

It appears that current smart cards are designed to protect the interests of banks, shops, and such rather than those of the customers. The large-scale automated transaction systems of the near future can (and should) be designed to protect the privacy and maintain the security of both individuals and organizations, as pointed out in [Cha85].

While waiting for the smart card technology to catch up with the requirements, it is possible to build larger trusted pieces of equipment that could, for example, be

integrated into a pocket calculator or a personal portable phone emerging with the Universal Personal Telecommunications (UPT) concept. In such a phone we would actually have a wireless connection to worldwide networks at our disposal where ever we are. On this scale, it is fairly easy to build tamper-proof modules to hold the critical parts of the system.

6.4 Real Cryptosystems

The core of this work is independent of existing cryptosystems and should remain that way. While the preceding chapters have not been based on any current cryptosystems, the availability of at least a good symmetric cryptosystem and a good asymmetric (public key) cryptosystem is assumed. To tie this work more firmly to the ground, some real cryptosystems, that can be used as pieces of the higher level mechanisms described earlier, are suggested here.

For a symmetric cryptosystem to be used for bulk encryption and calculation of cryptosums DES is suggested. While DES is not believed to be totally unbreakable, the cost of breaking it is still relatively high (probably even for NSA) and few parties are even suspected to have the skills required for a serious attack against it, other than exhaustive search of keys. DES is also reasonably efficient to execute with software and fast DES chips are in production. DES is described more closely in appendix 1 and in references [SB88, DP84, ANS81, ANS83].

For a Public Key Cryptosystem the Rivest-Shamir-Adleman algorithm (RSA) is recommended. With key lengths of about 1000 bits RSA is believed to be secure and it has some nice properties which make is well suited for generation of digital signatures as well as data encryption. Breaking RSA is as hard as factoring a large integer. Therefore, significant advances in the area of number theory could make RSA unusable. RSA is described more closely in appendix 1 and in references [RSA78, DP84].

7 Implementing Secure FTAM

The proof of a pudding is its eating.
(an old English saying)

Since the fall of 1989 implementation work based on the ideas presented in this study has been going on in Telecom Finland. As the first concrete application, a limited working prototype of secure FTAM is being built. The architecture of this project is described in [KHA90].

7.1 Requirements and Restrictions

The basic security requirements for FTAM were those stated in section 3.2.1, namely: Peer Entity Authentication, Connection Integrity, and Connection Confidentiality. It was concluded that Non-repudiation and Access Control could be left out for the time being.

7.2 FTAM Security Profile

The ultimate profile of Secure FTAM is presented in figure 7-1 below. The figure shows both communications and security profiles, security related parts of the profile are in bold-face. For practical reasons, the work was started with a modified profile and later extended to that of figure 7-1. Here the Secure FTAM Profile is presented briefly, layer by layer.

In the application layer, the latest versions of FTAM [IS8571-4] and ACSE [IS8650] protocols are used. In the first phase, authentication is done at the transport layer but it will later be moved up to the application layer. The Security Addendum of ACSE [IS8650/A1] is used for the two-way authentication exchange. A limited version of the SCSE (see section 3.4.3) is implemented in phase two to perform the actual authentication function.

In the presentation layer, the standard ISO presentation protocol [IS8823] is used. When selective field integrity and confidentiality are included in the SCSE (some time in the future), the presentation layer standards will have to be augmented with presentation contexts supporting data encryption.

In the session layer, the standard ISO session protocol is used. The functional units Kernel and Full Duplex are mandatory.

In the transport layer, the standard ISO transport protocol [IS8073], including operation on a connectionless network service [IS8073/A2], is used. Class 4 is supported, with negotiation down to class 0 if necessary. An augmented version of EESP [ISO90a] is used, where certain defects, ambiguities, and contradictions are solved in a way commonly agreed on in the European COST-225 Secure Communications project [CGV90].

In the network layer, the choice between two different subnetworks is given: one is a Local Area Network of the Ethernet-type, the other is the world-wide public packet-switched X.25 data network. Three alternatives, all in accordance with the principles of OSI sub-layering, exist: ISO IP [IS8473] can be run on top of ISO 8802-2 LLC [IS8802-2] or on X.25 PLP (CCITT 1984, [IS8208]); as the third alternative, X.25 can be used without ISO IP.

At the two bottom layers, either a local are network of the Ethernet type [IS8802-3] with ISO LLC type 1 [IS8802-2] or, with X.25, LAPB [IS7776] and X.21 physical interface [X.21] are used.

<table>
<tr><td>7. Application Layer</td><td colspan="2">ISO 8571-4
SCSE and ISO 8650/DAD1
ISO 8650</td></tr>
<tr><td>6. Presentation Layer</td><td colspan="2">ISO 8823</td></tr>
<tr><td>5. Session Layer</td><td colspan="2">ISO 8327</td></tr>
<tr><td>4. Transport Layer</td><td colspan="2">ISO 8073 and ISO 8073/Add. 2, Class 4 or 0
EESP</td></tr>
<tr><td>3. Network Layer (Internet)</td><td colspan="2">ISO 8473</td></tr>
<tr><td>(Subnet)</td><td></td><td>ISO 8208 (CCITT X.25)</td></tr>
<tr><td>2. Data Link Layer (LLC)
(MAC)</td><td>ISO 8802-2, Type 1
ISO 8802-3</td><td>ISO 7776 (LAPB)</td></tr>
<tr><td>1. Physical Layer</td><td>ISO 8802-3</td><td>CCITT X.21</td></tr>
</table>

Figure 7-1: Secure FTAM Profile.

Authentication is handled by the Secure Communications Service Element (SCSE, see section 3.4.3) with the help of the Association Control Service Element authentication facility [IS8650/A1]. Bulk Integrity and Confidentiality are achieved by using the End-system to End-system Security Protocol (EESP) [ISO90a] at the bottom of the Transport Layer, as proposed in sections 4.2 and 4.3. DES is used as the symmetric encryption mechanism required by EESP.

The EESP proposal also includes Peer Entity Authentication at the Transport Layer. It was decided, that the first working prototype should use the Peer Entity Authentication facility of EESP but based on asymmetric, rather than symmetric, encryption. At the next stage, authentication will be moved up to the Application Layer and based on ACSE, as proposed in section 4.1. RSA is used as the asymmetric encryption mechanism required for strong authentication.

7.3 Management Issues

The certificate distribution mechanism of the Directory [X.509] is used to distribute the public keys. The X.500 Directory User Agents (DUAs), located at the communicating end-systems, can query X.509-type certificates of the public keys of other parties from their Directory Server Agents (DSAs). The certified public keys are then used for mutual strong authentication and the exchange of a pair-wise session key to be used with the EESP.

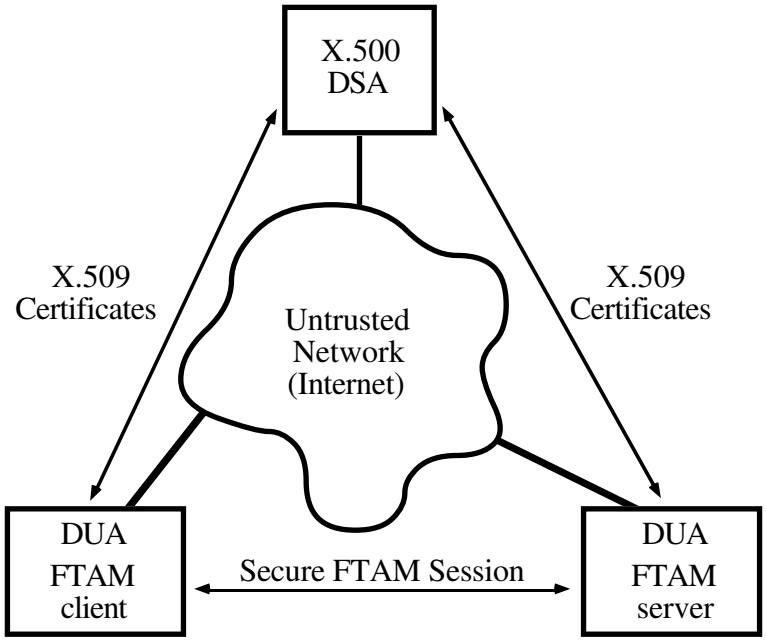

Figure 7-2: Key management in Secure FTAM.

7.4 Implementation Environment

The FTAM implementation of ISODE is used as the basis of this work for practical reasons. Because it is our intention to experiment with secure FTAM internationally within the COST-225 project, and because ISODE is the international de facto reference implementation it, rather than C-VOPS, was chosen as the basis of this work. A Sun 3/80 workstation connected to both a Local Area Network and the public X.25 Packet Switched Data Network is used. The system is first tested with TP0 running on both X.25 and TCP/IP. Later on, ISO IP will be used in the LAN.

The C-VOPS implementations of both the Directory Server Agent and the Directory User Agent of the X.500 Directory (including X.509) are used and run on Sun 3.

7.5 Current Status of the Project

The first version of Secure FTAM was successfully demonstrated at the COST-225 management committee meeting in Yugoslavia in October 1990. At that time, integrity, confidentiality, and primitive authentication (based on symmetric encryption and initial sequence numbers of PDUs), all done with EESP at the transport layer, were working.

At the next phase, public key encryption and certificates distributed via X.509 will be used for key management. Interfacing the key management with X.509 was well under-way in November. At the first stage, both the DUA and the DSA (see figure 7-2) are running on the same Sun 3 workstation and communicating via shared memory. Later on, a real, distributed directory system will be used. In the year 1991 the work will be continued by implementing Peer Entity Authentication in the Application Layer.

Due to some limitations of the latest ISODE version (v. 6.0), the current status of the system differs from figure 7-1 in several ways which are described here in some detail.

In ISODE, only class 0 transport protocol is currently supported on the LAN. Therefore, transport protocol class 4, which is the preferable class with connectionless networks, cannot be used in LANs. ISODE currently runs TP0 on top of TCP/IP, instead of ISO IP, as shown in figure 7-3 (just forget the EESP in end-system A and you see the current profile of ISODE).

TCP/IP offers quality of service comparable to that of the X.25-type packet switched networks, which makes the use of TP0 on top of it quite reliable. The solution is an intermediate one and, luckily, does not reflect to the upper layers in any significant way. Later on, the network layer can be changed into ISO IP and the transport protocol into class 4 without significant impact on the other layers.

The other alternative currently supported by ISODE is presented in figure 7-3 as end-system B (just forget the EESP again). This is already a pure OSI protocol stack, where class 0 transport protocol is run on top of the reliable virtual circuits offered by X.25.

In ISODE the LAN currently supported is not ISO 8802-3 (IEEE 802.3) but Ethernet 2. There are subtle differences between these two types of LAN, both in the physical and in the data link layer, which make them incompatible. Also, IEEE 802.2 Type 1, which really is functionally empty (just add a header consisting of a 0-octet to each frame), is not included in the current ISODE distribution. In fact, practically the whole world is using Ethernet 2 at the moment and even though stations of the two types can operate in the same physical network, they cannot (without clever tricks, such those used in the BSD 4.3 Unix) interoperate. Therefore, the use of Ethernet 2 is preferable for the time being.

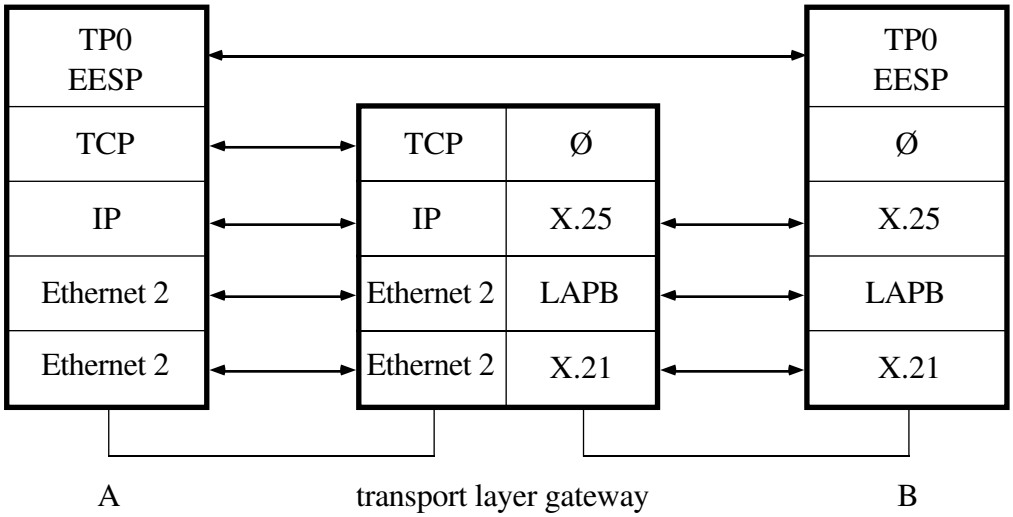

Figure 7-3: The current layers one through four.

As shown in figure 7-3, a transport layer gateway can be used to interconnect these two ISODE-based implementations in such as way, that EESP and TP0 truly are end-to-end. With this kind of gateway, which can easily be built with ISODE (and was already built and tested at the Technical Research Centre of Finland as early as 1986, see [TTY88], pp. 29-30), it is possible to communicate end-to-end between all the workstations connected to a LAN and all the computers connected to the world-wide X.25 public data network.

Since 1989 Telecom Finland has been offering the first public service for high-speed LAN interconnection in the world. The Datanet Service currently operates nation-wide at the maximum speed of 2 Mb/s, which will soon be significantly increased. For the time being only DoD IP is officially supported but support for other protocols, including the routing of ISO IP and tunneling of proprietary protocols, will be

announced soon. Infonet Inc. is developing a similar international LAN-interconnection service in cooperation with Telecom Finland. This means that the service will soon be available in most of Europe and the U.S. This is a clear indication of the validity of the speculation on network evolution presented in chapter 1.

In this kind of global, untrusted commercial network, the importance of open systems security cannot be over-estimated. As the leading teleoperator in Finland (where we have the most deregulated telecommunications business in Europe), Telecom Finland intends to be among the first teleoperators to offer secure services in a network with world-wide connectivity. The results of the implementation project of secure FTAM will provide valuable feed-back to the theory presented in this study and, possibly, lead to revising some parts of it. Being financed by Telecom Finland, it is only natural that this project should lead into commercial applications within a couple of years.

8 Conclusions

This work was based on the security requirements of real applications used in the business world and the real evolution of private and public networks. It was shown that tomorrow's world-wide interconnected networks cannot be trusted but secure information systems have to be developed to operate on an untrusted network. Therefore, physical security measures and cryptographic techniques within the networks cannot alone guarantee security in an open systems environment but end-to-end security needs to be built into terminal equipment (both terminals and computers) by using cryptographic techniques.

In this study, a security framework based on the properties of a layer entity was developed, integrated into the OSI reference model, and used to analyze a number of security functions and mechanisms and place them into the OSI model. The OSI reference model was taken as the basis of this study because it is the only universally accepted framework we have. However, a critical view was kept on the OSI model and several modifications and extensions to it were proposed. In this way, this study serves as a contribution to the ongoing, rather lively, OSI discussion. Most of this work is also directly applicable to other layered telecommunications architectures.

Starting from the customer needs, the security requirements of real applications were analyzed, prioritized, and seen to be, to a large extent, rather similar from one application to another. It was therefore concluded, that security should be an integral part of the quality of service provided by OSI rather than something implemented on a per-application basis.

A set of criteria affecting the placement of the security functions and mechanisms into the layers of OSI were presented. These criteria were then used to determine the proper place of each security function and mechanism in the OSI reference model. For each function, a mechanisms capable of implementing this function was found.

Management issues associated with these functions and mechanisms were studied and solutions were found to problems such as the activation of these functions and mechanisms and key management.

This work is not tied into any existing cryptosystems, which are expected to change continuously with the advances in cryptology (both cryptography and cryptanalysis). However, in order to tie this work more closely to reality, real cryptosystems, physical mechanisms for implementing them, and OSI software implementations, together forming a sound platform for real implementations, were pointed out.

The ultimate test of the ideas presented here is whether they will be found to be useful in implementing real secure open systems based, as much as possible, on the existing implementations. The solutions were tested in a pilot project implementing a secure version of the internationally standardized FTAM service. This work is still continuing, but encouraging results have already been achieved. Feed-back gained from this project will prove to be valuable in developing these ideas further.

It was shown that Secure Open Systems, in deed, are very much needed and that they can be efficiently built based on the existing protocols, cryptosystems, and OSI software implementations. This study provides an architecture and implementational guidelines for building secure open systems from these building blocks.

Along the way, several new concepts, such as *Security Context* and *Security Profile*, were defined and found to be useful. Many apparently fertile directions for further research in the field of Secure Open Systems were pointed out. The architectural framework and concepts developed in this study make it possible to formulate questions that have never been asked before, which brings a whole new research area, at the boundary between theoretical protocol research and cryptology, within the scope of formal research. In this way, this work opens more doors than it closes.

Among the directions for further work found and pointed out in this study are the following:

A formalism for analyzing the security of open systems and constructively building provably secure open systems needs to be developed as pointed out in section 3.4.

The security mechanisms described in the appendix need to be split up further in order to find the "atomic" elements of security. These elements appear to include mechanisms for en/decrypting messages and signing them. If these basic mechanisms can in turn be implemented on silicon with a reasonable amount of chip area they can be integrated into various ICs which constitute the basic physically secure "computers" interconnected via a global untrusted network consisting of hierarchy of networks (multichip carrier, circuit board, backplane, LAN, MAN, WAN) all with high bandwidth but the latency times increasing with the dimensions of the network.

If these two goals are reached we'll be able to construct provably secure fully distributed global information systems trusting only in the physically secure chips and our Certification Authority. The task should not be underestimated but this direction of research certainly looks very promising.

A1 Security Mechanisms

This appendix is provided as a general introduction to cryptology for those not familiar with the topic.

Security mechanisms, as defined in [IS7498-2], are the means by which the OSI Security Functions can be provided. In the OSI Security Architecture mechanisms of quite different levels are all called security mechanisms. In fact these mechanisms form a hierarchy:

- *Higher level mechanisms*, such as security protocols and (semantic) message contents.

- *Lower level mechanisms*, such as cryptosystems, forming parts of the above-mentioned higher level mechanisms.

- *Physical mechanisms*, such as encryption chips and pieces of program code, implementing the above-mentioned mechanisms.

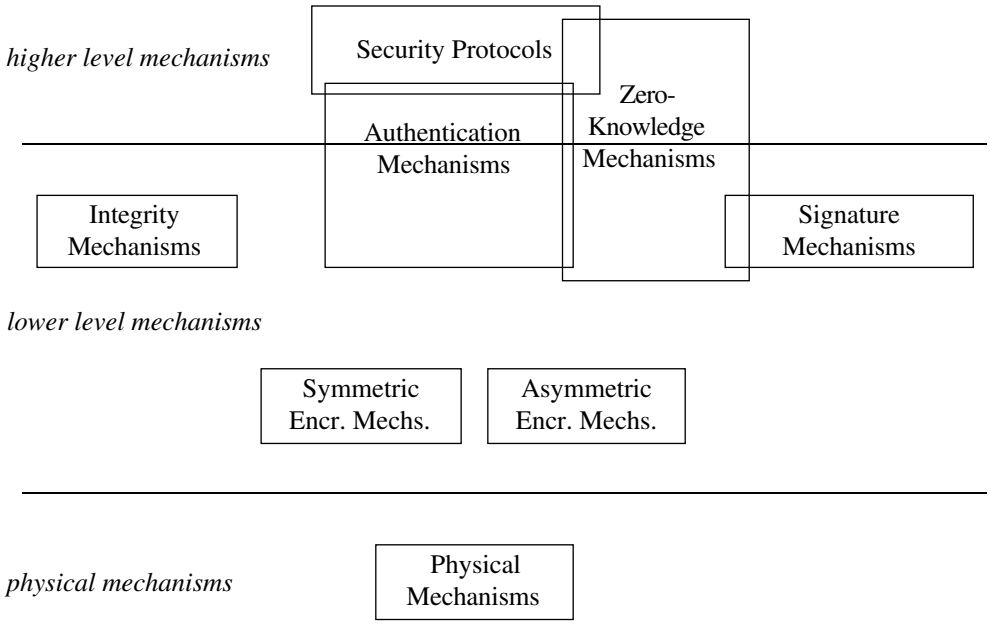

Figure A1-1: The hierarchy of security mechanisms.

The security of a local system can, to a great extent, be ensured by physical security arrangements. In a global untrusted network it is impossible to guarantee the security of communications by means of physical security but cryptographic techniques are needed. A lot of work has been done in the area of cryptology during the past couple of

thousand years but the most significant progress has taken place after the advent of electronic computing. Today we have at our disposal a number of cryptographic mechanisms that can be used in connection with open communications architectures.

Among the basic cryptographic mechanisms are those used for data encryption, integrity and authentication. Data encryption mechanisms can be used for all these three purposes and they are usually divided into symmetric and asymmetric cryptosystems. There are cryptosystems, not based on asymmetric encryption, that can be used for purposes or authentication and non-repudiation. Here we shall discuss the four main classes of cryptographic mechanisms: *symmetric encryption mechanisms*, *asymmetric encryption mechanisms*, *signature mechanisms*, and *integrity mechanisms*, separately. The use of signature mechanisms for *authentication* is discussed and the important class of *zero-knowledge techniques* is briefly dealt with. Finally, the implementational aspects of cryptographic mechanisms and the *physical mechanisms* implementing them are touched upon.

While this chapter is outside the core of this study, it is necessary to discuss security mechanisms in some detail in order to demonstrate the implementability of the security architecture proposed and tie this work more firmly to reality.

A1.1 Data Encryption Mechanisms

An *encryption mechanism* is used to convert a cleartext message into a cryptogram, that is into a form that is unintelligible to anyone not knowing the secret decryption method. An encryption mechanism is based on a *public algorithm* and at least one *key* whose value is randomly chosen from a large set.

The need for separating the algorithm from the keys is obvious. In this way we can use the same (hardware or software) implementation of a cryptosystem over and over again with different key values. The algorithm remains the same and only a relatively compact key value needs to be changed.

The reason why the algorithm needs to be public is less obvious. The security of a cryptographic mechanism should never be based on the secrecy of an algorithm. This is a generally accepted principle known as the *Kerckhoff's principle* (named after a 19th century Flemish linguist). Only an algorithm that has been open for public criticism and cryptanalysis for a reasonable period of time can be trusted to be secure. Secret algorithms may (intentionally or not) contain trapdoors and other faults.

In fact, it can be claimed that not only the security of a cryptosystem but, in deed, the security of the entire open system should always be based on public mechanisms and secret keys. That is, everything else about a secure open system except the key values

should be public and open for analysis and criticism. Not only is this the only way to gain reasonable assurance of the security of an open system but it also seems to be a prerequisite for any legislation related to data security giving the security mechanisms used a legal standing (e.g. making digital signatures binding in the court of law).

An interesting current research area in cryptology is that of *elliptic curve cryptosystems* based on a group of points on an elliptic curve. Originally proposed in 1985 by V. Miller of IBM and further developed by N. Koblitz and others, these mechanisms seem to promise public key cryptosystems more secure than RSA with comparable block lengths in the near future. These cryptosystems are believed to be computationally intensive but the current fast progress in VLSI technology will help in their efficient implementation [BVA90].

This is not meant to be a study on new emerging cryptosystems or their principles. The only motivation here is to show, that we cannot expect anything of the nature of the keys of possible future cryptosystems. They may, for example, be sets of data items or functions. Therefore, it is vital that we leave enough flexibility in our current designs to accommodate for new cryptosystems which inevitably will emerge. In this study the word *key* is used to denote any (secret or public) reasonably compact piece of information which can be presented digitally (as a bit-string) and used in connection with a public algorithm to form a cryptographic function.

A1.1.1 Symmetric Encryption Mechanisms

In a *symmetric cryptosystem* we have two functions *encrypt* and *decrypt*. A message encrypted with key K can only be decrypted with the same key. Symmetric key cryptography, which dates back to ancient times, was developed into exact science in Shannon's famous paper written in 1945 as a secret report and declassified and published in 1949 [Sha49].

From here on we shall use the following notation for encrypting with a symmetric cryptosystem: c = eK(m), where m is the *cleartext message*, K is the *key*, e is the *encrypt function* and c is the *cryptogram*. Similarly, we use the following notation for decryption: m = dK(c), where d is the *decrypt function*.

In our notation, the key used is combined with the function rather than supplied as one of its arguments in order to point out the fact that the public part of the function and the key together form a unique function used for encryption or decryption. The key can be thought of as an index choosing one of a family of functions.

A1.1.2 Asymmetric Encryption Mechanisms

The first mention of *public key cryptography* is found in [DH76]. The first implemented *public key cryptosystems* (from here on simply *PKCs*) were the Rivest-Shamir-Adleman (RSA) scheme, presented in [RSA78], and the Merkle-Hellman *trapdoor knapsacks*, presented in [MH78]. For an introduction to the history and current status of public key cryptography see e.g. [Dif88] and [BVA90]. A short introduction is also found e.g. in [Den83]. Even though the original Merkle-Hellman knapsacks were soon broken, some of their improved versions are still believed to be quite secure. Some improved Merkle-Hellman knapsacks are presented e.g. in [LLH89].

In a (PKC) the key can be divided into two parts, the *encryption key* and the *decryption key*, in such a way that the encryption key specifies the *encryption transformation* and the decryption key determines its left inverse mapping *decryption*. If it is *computationally unfeasible* (for a more precise definition of the term see [DH79]) to determine the decryption key from the encryption key the PKC is called an *asymmetric* (or *public key*) *encryption mechanism*. If it is unfeasible to derive the encryption key from the decryption key the PKC is called a *public key signature mechanism*.

An asymmetric encryption mechanism provides complete confidentiality (only the legitimate recipient in possession of the secret key can decrypt the message) but no authentication of the sender (anybody with access to the recipient's public key could have generated the message).

From here on we shall use the following notation for encrypting with the public key: $c = eK_{P,X}(m)$, where m is the cleartext message, $K_{P,X}$ is the public key of X, e is the Encrypt function, and c is the cryptogram.

The following notation is used for decrypting with the secret key: $m = dK_{S,X}(c)$, where $K_{S,X}$ is the secret key of X and d is the *decrypt* function.

A1.1.3 On the Use of Encryption Mechanisms

For encrypting large amounts of data symmetric cryptosystems are preferable to asymmetric cryptosystems e.g. for the following reasons:

Firstly, symmetric encryption is much more efficient than asymmetric encryption. For example, both hardware and software implementations of DES are currently about 1000 times faster than corresponding RSA implementations (with a reasonable key length).

Secondly, when multicasting a large message it is much more efficient to use symmetric than asymmetric encryption. If the message is encrypted with the public key of the

recipient, a different key has to be used for each recipient, causing the same message to be encrypted and transferred several times. With symmetric encryption, only the symmetric key used needs to be encrypted separately and passed to each recipient, then this one key (now a part of the security context associated with this message) can be used to encrypt (and decrypt) the data which is multicast to all of the recipients. This scheme is adopted e.g. in the *Privacy Enhancement for Internet E-mail* [RFC89a,b,c].

A1.2 Signature Mechanisms

Digital signatures can be used to realize various security functions, including *authentication* (both of *peer entity* and *data origin*), *non-repudiation* (both of *origin* and *delivery*), and *integrity*.

The digital signature of a message is a redundant piece of information which depends on the entire contents of the message in such a way that only the party in possession of the secret signing function can produce it but its validity can be checked without this secret information. The signature (when associated with the message) proves both the origin and integrity of the message providing the function of Non-repudiation of Origin.

Just like with encryption mechanisms, a good digital signature scheme has the property that it is based on a public algorithm and a secret key. Public information is used for checking the signature. It should be computationally unfeasible to generate valid signatures without knowing the secret key as well as to find two different messages with the same signature value.

A PKC is a signature mechanism if the encryption or *signing*, as we may call it now, is secret and the decryption, i.e. the checking of the signature is a public algorithm. When using RSA, the most straight-forward way of signing a message is to encrypt the whole message with the secret key of the sender. In this case we need not send the cleartext message at all. If the message has sufficient redundancy, so that a randomly chosen cryptogram does not decrypt into an acceptable cleartext message with anybody's public key, we can be confident in, and show to an impartial judge, that the message was originated by the holder of the secret key. This kind of signature scheme is presented in [IS9796] *Signature Algorithm with Message Recovery*, which currently is a DIS version.

A more common practice is to send the cleartext message together with the signature. This makes it possible to read the contents of the message at intermediate nodes without having to decrypt the message.

For performance reasons we usually encrypt a hash value based on the message rather than the whole message. The hash function used in computing the hash value has to have certain properties described more closely in e.g. [DP84]. Obviously we must claim the hash function to be such, that it is computationally unfeasible to find two valid cleartext messages with the same hash value.

Hash functions are currently an area of very active research. A proposal for a hash function which is light to compute with software is presented in [Riv90]. Among other recent proposals are Merkle's Snefru function and Miyaguchis's N-hash [MOI90]. Biham and Shamir have shown in [BS90] that it is easy to find synonyms (i.e. different messages with the same hash value) for all of these three algorithms. Also in ISO standard hash functions for digital signatures are being defined [IS10118-1,2]. [Mer89] discusses the use of DES in computing one-way hash functions. DES-based hash functions still appear to be a safe choice for most commercial purposes.

A message is now signed in three steps:

- Calculate the hash value of the message.

- Encrypt the hash value with the secret key of the sender.

- Attach the encrypted hash value (signature) to the original message.

The notation $S_X\{m\}$ is used in this study to mean "message m signed by X". The notation does not stand for the signature but for the message together with its associated signature, hence the notation (use of curly brackets instead of parentheses). The notation $S_X(m)$ is used to denote the signature value of message m generated by X. This practice is consistent with that adopted in e.g. [X.509].

A1.3 Integrity Mechanisms

Data Integrity can be divided into two parts:

- *Message* (or *content*) *integrity* means that each individual message is intact (that is, received in exactly the same form as sent).

- *Sequence integrity* means that the sequence of messages (usually the sequence of messages sent across a connection during its life-time) is intact. This means that no messages have, undetected, been omitted or duplicated and that the original ordering of the messages is preserved.

In [ISO90d] integrity mechanisms are classified as those designed to protect against random modifications and those designed to protect against modifications deliberately engineered to defeat the integrity mechanism. In this study, we shall use the terms *weak* and *strong integrity*, respectively, for these two cases. Data integrity in the weak sense of the word means detecting all incidental (random) changes in the data being transferred, optionally recovering from the changes, when possible, and reporting the cases where recovery is not possible. Weak integrity is normally provided at the Data Link layer point-to-point (e.g. HDLC) and at the Transport Layer end-to-end (e.g. ISO TP4). Protocols of both of these layers are designed to detect and recover from transmission errors (but not planned attacks against data integrity).

For weak message integrity simple redundancy in the data is adequate. Usually a Checksum or, preferably, a Cyclic Redundancy Check (CRC) is used to detect chances in the data. In LANs a 32-bit CRC is normally used and found to be quite effective for this purpose.

Sequence integrity is achieved by labelling the messages with a running sequence number which can be used to detect omitted, duplicated, or out-of-sequence messages. To detect messages omitted at the end of a connection, a graceful disconnect procedure with the negotiation of the last sequence number sent is required before closing the connection.

For purposes of error recovery, retransmission of corrupted messages is used (Automatic Repeat Request, ARQ). In time-critical applications or with unidirectional transmission systems, error-correcting codes, such as Hamming codes, are sometimes used (Forward Error Correction).

Integrity in the strong sense of the word means detecting not only accidental errors but also those deliberately caused by an enemy by means of an active attack. Here simple redundancy added to the messages with public algorithms is not sufficient to guarantee message integrity and the use of sequence numbers is not sufficient to guarantee sequence integrity because both can easily be forged.

In the set-up illustrated in figure 4-2, X can easily modify the data stream between A and B, without this being detected by either communicating party, if only weak integrity is provided. A simple way of doing this is by letting an untrusted intermediate node in the network, such as a router, act as a transport layer gateway between A and B, using the standard transport protocol towards both A and B, relaying the information transferred from one connection to another, modifying the information at its will, and letting both A and B believe that they are communicating end-to-end at the transport layer. The attack is illustrated in figure 4-2 below.

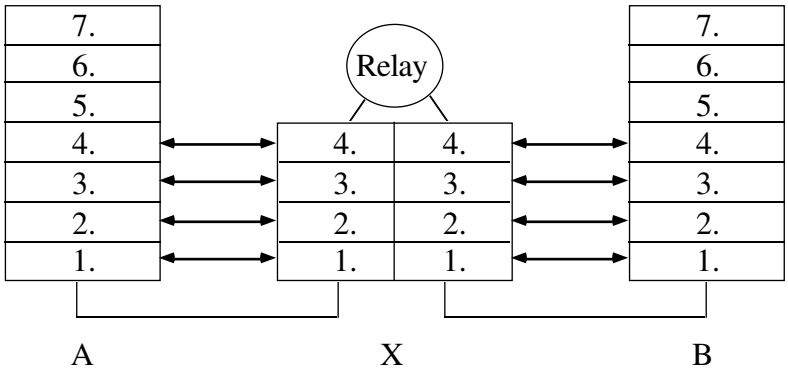

Figure A1-2: Attack against a weak integrity mechanism at the transport layer.

Strong Integrity can be efficiently achieved by using the orthogonal function of *confidentiality* in combination with a *weak integrity mechanism*. If the confidentiality function is placed below the weak integrity mechanism in the OSI Protocol Stack, then the result is strong integrity. This result can be justified with the following reasoning:

- Assuming that the encryption mechanism is an acceptable one and that the enemy does not possess the key used, it is impossible for the enemy to determine anything of the contents of the cleartext message by studying the cryptogram. It is equally impossible for him to generate a cryptogram that would decrypt into anything but a random cleartext message.

- Therefore, from the weak integrity mechanism's point-of-view, any changes introduced by the enemy to the stream of cryptograms are seen as completely random changes in the stream of cleartext messages. But these random changes are just what the weak integrity mechanism was designed to protect against.

It must be noted that there are a number of requirements for an "acceptable" encryption mechanism and weak integrity mechanism as well as for their combination. For example, when a block cipher (with the ECB mode of operation) is used, the weak integrity mechanism must be able to detect errors in the ordering of the encrypted blocks.

If, for example, all TPDUs of TP4 are encrypted, then the Integrity Check Value carried by each TPDU is also encrypted and the TPDUs, including their sequence numbers, cannot be modified by the enemy without this being detected. This makes all attempts to modify the content of individual messages or the sequence of messages rather random and enables the normal functions of TP4 to detect (and attempt to recover from) these modifications.

The situation is clarified in figure 4-3 below. In case a) TP4 is run on top of a noisy channel. This results in reliable transfer on the transport connection between the two Transport Service Access Points. In case b) the channel is not only noisy but hostile, meaning that it causes planned non-random changes in the data stream transferred (e.g. the situation depicted in figure 4-2). This results in unreliable transfer on the transport connection. If end-to-end encryption is introduced below the transport protocol, as depicted in case c), all changes caused to the data stream by enemies not in possession of the encryption key are random and result in the illusion of a noisy channel at the interface between the encryption layer and the actual transport layer. This reduces the situation to that of case a) resulting, again, in reliable transfer on the transport connection.

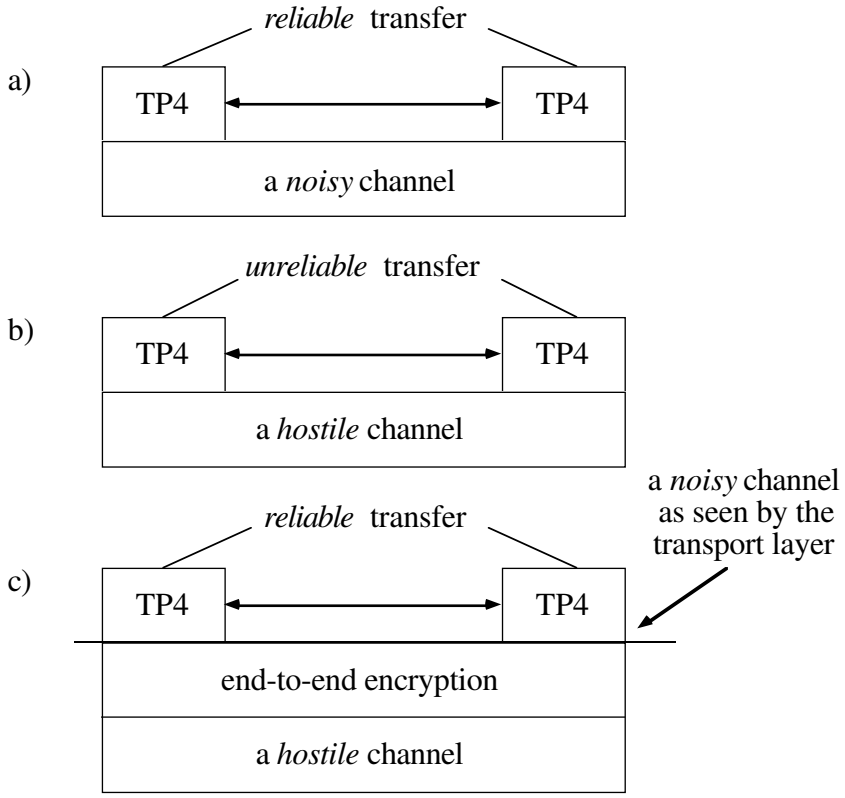

Figure A1-3: Weak integrity combined with encryption produces strong integrity.

In order to prove the reasoning presented above for a given transport protocol – data encryption scheme pair, detailed requirements of the transport protocol (e.g. TP4) to the underlying "channel" (network service) in order to guarantee reliable transport service must first be stated. Then it must be shown, that the hostile channel augmented with the proposed encryption scheme (e.g. DES) does fulfill these requirements. Because the purpose of this work is to create an architectural framework, not to analyze the properties of individual cryptosystems (such as DES) or protocols (such as TP4), the detailed analysis is left for further study.

This revelation gives us one more criterion for placing the functions and mechanisms of Integrity and Confidentiality in the layers of OSI. To avoid unnecessary duplication of functionality, it usually seems to be a good idea to place a confidentiality mechanism somewhere below an existing weak integrity mechanism and get strong integrity as a by-product of these two mechanisms.

S. Walker proposes, in [Wal89], that EESP-like protocols should be placed *above* rather than below the normal transport protocol in order to minimize the overhead caused by encryption and decryption in case of error recovery at the transport layer. It is my belief that efficiency under normal operation is by far more important than efficiency in the rather rare case of retransmission at the transport layer. Obviously this also depends on the quality of the underlying network service.

We can determine that, with some restrictions, a reliable *weak integrity mechanism* when used on top of a reliable *confidentiality mechanism* provides us with the function of *strong integrity*. The problem of which encryption mechanisms are "acceptable" and which weak integrity mechanisms they should be combined with in order to produce strong integrity is a current research topic of cryptography. One basic requirement appears to be that the integrity check value employed by the weak integrity mechanism should be non-linear. It is generally known that block cipher with the OFB mode of operation together with linear redundancy in the cleartext message (for example DES with the OFB mode of operation in connection with Hamming coding) does not produce strong integrity. Integrity mechanisms are also being standardized by ISO. [IS9797] is an example of a standardized strong integrity mechanism which is currently a draft proposal.

A1.4 Authentication

Authentication is divided into two classes:

- *Weak* (or *simple*) *authentication*, where a password but no encryption may (or may not) be used, and

- *Strong authentication*, based on cryptographic techniques.

Weak authentication offers no protection (if not even passwords are used) or inadequate protection against malicious attacks. Weak authentication is currently used in most systems (e.g. practically all operating systems) but, even with passwords, it is very vulnerable to replay. In strong authentication, the user (or service) can prove his identity without revealing his secret piece of information. The need to authenticate a user without him having to reveal his secret authentication information is pointed out

e.g. in [Sim89] where a simple identity verification scheme is presented providing unforgeable transaction receipts for later arbitration.

A requirement for any type of trust in a verifier by a claimant should not be a pre-requisite for the correct operation of any authentication mechanism. If passwords are used, then the claimant always has to trust the verifier not to keep and reuse his password to impersonate as him to another entity [IS10181-2]. This observation alone is sufficient for ruling out the use of passwords as an acceptable general solution to authentication. It has also been shown (e.g. in [MT79]) that password-based authentication schemes can never be made completely reliable. Therefore, in this study, only strong authentication is discussed, since weak authentication clearly does not offer adequate security for our needs.

Strong authentication can be based on symmetric or public key cryptography. With symmetric cryptosystems a mutually agreed-upon pairwise key belonging to the appropriate security context is needed for authentication between any pair of parties A and B. Therefore, with n communicating parties n(n-1)/2 keys are needed. For large values of n this is clearly impractical. The number of keys required is too large to be securely generated, transferred and stored, every party having to be in possession of n-1 pairwise keys.

Public key signature mechanisms have several advantages over symmetric cryptosystems when used for authentication. Key management is greatly simplified by the fact that only public keys of the key pairs need to be shared and only one key pair is needed for each party. Also, since the secret key is only known to one party, public key signatures can be used for non-repudiation.

Even though symmetric encryption schemes can be used for authentication they cannot easily be used for the stronger service of non-repudiation. Because a symmetric communication key always has to be shared by at least two parties, it is impossible to prove to an impartial judge the origin of a message signed with an algorithm based on symmetric encryption without relying on a trusted server of some kind, such as an on-line notary. If A produces a message signed with a pairwise key between A and B, claiming that this message was created by B, B can always claim that the message was forged by A, A also being in possession of the same key. In a case such as this, only A and B know the truth but neither can convince the judge. Some signature schemes based on symmetric encryption are presented in e.g. [Akl83] and [Dav83] but they are rather artificial (i.e. they "consume" keys when used) and not very well suited for our purposes.

From the authentication point-of-view we simply need a trusted certification authority whose certificates can be used to verify the integrity of other parties' public keys. The way in which these certificates are distributed is a management issue. The X.500

Directory Service (more precisely X.509) provides us with a natural means of distributing these certificates. It is worth pointing out (again) that the directory itself need not be trusted, we only need to trust our CA.

Authentication Protocols

Authentication protocols have been widely studied (see e.g. [NS78, DS81, NS87, DH79, Moo88, IM90]). Often the need for three-way handshake for reliable mutual authentication has been pointed out. The main motivation for a three-way handshake mechanism is the need for each party to challenge the other.

The mechanisms included in the ACSE Security Addendum can be used to implement the authentication procedure describe above. This arrangement allows the client (initiator) to challenge the server (responder) by having him increment the time stamp by one. However, the server does not get to challenge the client. This is usually considered a shortcoming in ACSE's authentication mechanism, since three-way handshake was ruled out of ACSE in ISO.

The idea of exchanging challenges is that in this way each party can make sure that the authentication message received from the other party was not a replay. However, it is not the exchange of challenges per se but rather the *freshness* of the tokens that is required. With the freshness of a token we mean that it was generated for this purpose and has not been used before. It is shown below that two-way handshake is adequate for reliable mutual authentication when used with a time stamping mechanism. The principles presented here are nowadays widely known and accepted, the interested reader is referred to e.g. [IS9798-1,2,3] and [IS9799]. However, a brief summary of the principles of authentication based on a two-way handshake protocol is given here.

Assume that:

1) Each authentication message from A to B includes $S_A\{A, B, t\}$ (sender, recipient and time stamp, signed by the sender).

2) B is responsible for storing all authentication messages still valid (those, whose time stamps have not yet expired) and detecting attempted replay.

3) The signature scheme is reliable (only A can produce an acceptable message signed by A).

Conclusion: two-way authentication is adequate, because:

1) Message $S_A\{A, B, t\}$ is only accepted by B (the indicated recipient).

2) It is only accepted if it is fresh (generated by A for this purpose), i.e. the time stamp is still valid and B has not received the same message before.

3) B knows that the message was *for him*, *from A* wishing to authenticate *now*.

4) Likewise, A knows that the response from B: $S_B\{A, B, t+1\}$ was *for him*, *from B* and a reply to his request.

The fields for sender and recipient (A and B) included in the signed portion prevent the use of a captured message anywhere except between A and B. The time stamp values (t and t+1, where the time unit is the resolution of the time stamp used, i.e. one "tick") link the two authentication messages together and, when combined with the obligation of B to detect replay, make up for the lack of challenges chosen by A and B. The order of A and B (not implied with this notation) prevents the use of a response as a request later. These conditions appear to be sufficient for the conclusion that A and B unquestionably have mutually authenticated.

These conditions are thought to be sufficient but not necessarily minimal. As demonstrated by the error in X.509 [BAN89, IM90] the strive towards minimality can be dangerous, and it is by no means very useful. The conditions stated above are, in any case, reasonable and can be fulfilled without excessive over-head.

In case public key signature mechanisms are used for purposes of authentication (which most often is the case), it is necessary to include the recipient in the signed portion of the message to avoid the same message being used for another purpose later (referring, again, e.g. to the X.509 attack in [BAN89]). However, if symmetric cryptosystem and pairwise keys are used for authentication, it is possible to do without the Sender and Recipient fields in the signed portion of the message. That is because a given authentication key is only valid between a certain pair of communicating parties (such as A and B) and replay is impossible outside this Security Context. Here too, it must be assured that a reply cannot be used as a request later. With public key cryptosystems authentication message originally sent from A to B could later be used by B trying to impersonate as A to C if the recipient was not included. The necessity of the sender field can always be argued, it is included mainly for consistency.

The only question left open is whether, after the authentication exchange, both A and B know that a connection has been opened and a session key (completing the common Security Context) has been agreed upon. Obviously, after two-way handshake, B cannot know that A has received and accepted his last message. The problem is analogous to that found in establishing a transport connection. Also the solution is the same.

We can require that A sends the first (encrypted) message on the newly established connection as soon as it receives the response from B. Authentication is only considered completed after one more message is passed from A to B. This third message, after the two previously mentioned authentication messages, concludes a kind of three-way handshake. If B does not receive this message within a reasonable time, a time-out occurs and the connection is aborted.

A more formal proof of the adequacy of this two-way authentication protocol could be given by using the "Logic for Authentication" presented in [BAN89]. However, this logic is still at its early stage (as pointed out e.g. in [Nes90]) and does not (yet) fulfill the requirements for a serious mathematical formalism. Substantial extension to the BAN logic are proposed in [GNY90]. It is also used in [CG90] to analyze a secure multi-party session protocol. While it is evident that this kind of formalism is needed in developing and analyzing security protocols, a ready-to-use formalism does not currently exist and developing one is a major undertaking and clearly beyond the scope of this study.

The authentication function per se does not protect against an untrusted party within the network intercepting the connection later, after authentication, and impersonating as A to B and vice versa. However, if a pairwise symmetric session key is securely exchanged between A and B during the authentication exchange and used in subsequent communication, e.g. in the way presented in figure 4-3, this threat can be protected against (an authentication protocol that enables us to securely exchange secret information in connection with the authentication exchange is called a *subliminal channel*). This session key can be generated by either of the communicating parties alone or both together.

The communicating parties can authenticate each other (both at the end system level and at the user level) in the following manner. The process is initiated by user U logging in a service. The initial log-in is omitted in the following description, which concentrates on the security related aspects of the communications sequence.

The application entity A_U serving user U forms token $T_U = \{U,A_U,S,A_S,TD\}$ and signs it with its public key. If authentication of the user is required, the message is handed over to be signed by U. A_U then sends the signed token over to A_S. A_S (and S) can now authenticate both U and A_U by checking that the signatures are intact and the time stamp is valid. This procedure can be performed without U giving its secret key to A_U or vice versa.

In a similar fashion, the server S and its application entity A_S can identify themselves to U and A_U without trusting each other with their secret keys. This makes it possible to use, for example, a smart card for authentication of the user.

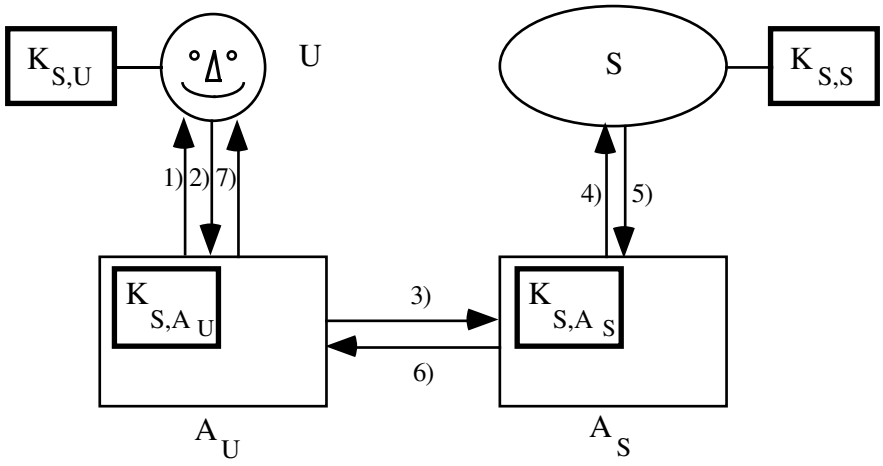

Figure 4-4: Strong two-way authentication.

Strong two-way authentication goes as follows:

A_U generates token T_U={U,A_U,S,A_S,TD}, where TD is a time stamp, and passes it on to U:

1) T_U

U signs T_U (with its secret key $K_{S,U}$) and responds with:

2) $S_U\{T_U\}$

A_U co-signs the result (with its secret key K_{S,A_U}) and sends the signed message over to A_S:

3) $S_{A_U}\{S_U\{T_U\}\}$

A_S checks the signature of A_U, thereby verifying its identity. A_S also checks the freshness of T_U by checking that the time stamp TD is still valid and (if it is) that the same token has not been received before. A_S now generates token: T_S={S,A_S,U,A_U,TD+1} and passes it on to S together with T_U signed by U:

4) $\{T_S, S_U\{T_U\}\}$

S checks the signature of U, signs T_S (with its secret key $K_{S,S}$). S can also generate a pair-wise fresh session key, encrypt it with the secret key of U, and include it in T_S before signing, yielding: T_S={S,A_S,U,A_U,TD+1,e$K_{P,U}$(K_{AB})}, as described in appendix 3. S now responds with:

5) $S_S\{T_S\}$

A_S co-signs the result (with its secret key $K_{S,AS}$) and sends the signed message over to A_U:

6) $S_{AS}\{S_S\{T_S\}\}$

A_U checks the signature of A_S, thereby verifying its identity. A_U also checks that the value of the time stamp in T_S is TD+1 (where TD is the time stamp of the corresponding T_U) and that the time stamp is still valid (not too much time has elapsed). A_U then passes T_S signed by S on to U for it to check the signature:

7) $S_S\{T_S\}$

U checks the signature of S completing the authentication cycle. If T_S includes the encrypted pair-wise session key $eK_{P,U}(K_{AB})$, U recovers the key by decrypting it with its secret key $K_{S,U}$.

Confidential encrypted information can be included in the signed portions of the authentication messages. One way of doing this is by using a PKC and the key pair of U (the calling party) in message 6), as describe above and in appendix 3. In this case the pairwise session key is generated by the service, which usually is a good approach when using a smart card with limited memory and processing capacity for user authentication.

It is also possible to have the calling party U generate the session key. In this case the PKC key pair of S (the called party) is used in message 3). In message 6) either a PKC and the key pair of U or a symmetric cryptosystem and the session key included in message 3) can be used for confidentiality. The latter approach has two benefits:

- By using the pairwise key included in message 3) S acknowledges this session key and U can now be sure that S agrees on the use of this key.

- Using a symmetric cryptosystem is computationally more efficient than using a PKC.

If a fresh, pairwise session key is securely exchanged in connection with the authentication procedure, we can be confident that only the two mutually authenticated parties are in possession of this key. This key can then be used for purposes of continued authentication, integrity and confidentiality over this session, e.g. in the way illustrated in figure 4-3.

Two considerations regarding the exchange of a fresh session key in connection with the authentication exchange are worth pointing out:

- The session key must be signed by the sender, that is it must be included in the portion of the message affecting the value of the signature.

- Even though the session key is always sent in an encrypted form, the signature function should preferably be performed before the encryption, because signing encrypted data is always considered a bad practise as shown e.g. in [MWR89] or [IS9798-3].

The authentication protocol described above can be realized by using a symmetric block cipher (such as DES) and a PKC (such as RSA) in the following way:

- The symmetric cryptosystem is used to calculate the hash function h (e.g. with Cipher Block Chaining mode of operation).

- The PKC is used for encrypting and decrypting the hash values. For this, the encryption key is kept secret while the decryption key is made public.

The application entity A_U serving User U forms token $T_U = \{U,A_U,S,A_S,TD\}$ and signs it. A_U then sends the signed token to A_S. A_S and S can now authenticate U and A_U, respectively, by checking that both of the signatures match and the time stamp TD is valid. Mutual authentication is achieved by having A_S and S perform the symmetric operation.

Mutual strong two-way authentication now proceeds as follows (see also figure 4-3):

A_U generates token $T_U = \{U,A_U,S,A_S,TD\}$ and passes it on to U:

1) T_U

U signs T_U, by calculating $h(T_U)$, encrypting this with its secret key $K_{S,U}$, and appending this signature to T_U, and returns the signed token to A_S:

2) $\{T_U, eK_{S,U}(h(T_U))\}$

A_U co-signs T_U by encrypting the signature of U with its secret key $K_{S,AU}$: $X_U = eK_{S,AU}(eK_{S,U}(h(T_U)))$ and sends this signature X_U, together with T_U, over to A_S.

3) $\{T_U, X_U\}$

A_S decrypts the signature with the public key of A_U recovering the signature of U. A_S also checks the freshness of T_U by checking that the time stamp TD is still valid and (if it is) that the same token has not been received before. A_S now generates token: T_S = {S,A_S,U,A_U,TD+1} and passes it on to S together with T_U and the signature of U:

4) $\{T_S, T_U, dK_{P,U}(X_U)\}$

S checks the signature of U by decrypting $dK_{P,U}(X_U)$ with the secret key of U and comparing the result with $h(T_U)$. S now signs T_S by encrypting $h(T_S)$ with its secret key $K_{S,S}$, appends this signature to T_S, and passes the signed token back to A_S. Optionally S can generate a pair-wise session key K_{AB}, encrypt it with the public key of U, and append it to T_S before signing (this seems to be one of the rare cases, where signing encrypted data is justified). This means substituting T_S with $\{T_S,eK_{P,U}(K_{AB})\}$ in the message below (the optional part is denoted with angle brackets):

5) $\{\{T_S\ [,eK_{P,U}(K_{AB})]\}, eK_{S,S}\{h(T_S\ [,eK_{P,U}(K_{AB})])\}\}$

A_S co-signs T_S by encrypting $eK_{S,S}(h\{T_S[,eK_{P,U}(K_{AB})]\})$ with its secret key $K_{S,AS}$: X_S=$eK_{S,AS}(eK_{S,S}(h\{T_S[,eK_{P,U}(K_{AB})]\}))$ and sends the resulting signature X_S, together with T_S, over to A_U:

6) $\{\{T_S\ [,eK_{P,U}(K_{AB})]\}, X_S\}$

A_U checks the signature of A_S, by decrypting X_S with the public key of A_S and recovering the signature of S: $eK_{S,S}(h\{T_S[,eK_{P,U}(K_{AB})]\})$. A_S also checks that the value of time stamp in T_S is TD+1 (where TD is the time stamp of the corresponding T_U) and that the time stamp is still valid (not too much time has elapsed). A_S now passes T_S (together with the optional encrypted pair-wise session key) and the signature of S and on to U:

7) $\{\{T_S\ [,eK_{P,U}(K_{AB})]\}, dK_{P,AS}(X_S)\}$

U checks the signature of S by decrypting it with the public key of S and verifying that $h(T_S[,eK_{P,U}(K_{AB})]) = dK_{P,S}(dK_{P,AS}(X_S))$, completing the authentication cycle.

The key exchange scheme presented here is by no means the only possible way of securely exchanging a fresh, pair-wise session key in connection with the authentication exchange. It is, for example, possible to have U (instead of S) generate the key or exchange the key on application (rather than user) level.

However, the scheme presented here appears to be well suited for user identification based on a smart card. Here the burden of key generation is left entirely to the service

which has better facilities for this purpose. Also, if the smart card has an integrated symmetric encryption mechanism, the key can easily be loaded onto the smart card and decrypted there without it ever leaving the card in clear-text form.

A1.5 Zero-Knowledge Techniques

A rapidly growing area is that of zero-knowledge techniques. In these techniques, the secret authentication information of each party plays very much the same role as the secret key in PKCs but it cannot be used for data encryption, only for authentication (and possibly digital signatures). In some zero-knowledge schemes, unlike in PKCs, the "public key" needed to verify an entity can be the same for all entities belonging to the domain of the same CA. Another typical characteristic of current zero-knowledge schemes is, that they often use several iteration cycles of challenges and replies to them to achieve the required level of confidence in the identity of the other party.

A class of authentication schemes, where the the *claimant* can prove his identity to the *verifier* without revealing a single bit of his secret authentication information, is known as Zero-knowledge techniques. Perhaps the nicest property of Zero-knowledge techniques is that they eliminate the need to periodically change the authentication information in order to protect against cryptanalysis, because the verifier (or a casual eavesdropper) cannot learn anything in the process that would help him masquerade as the claimant later. The first zero-knowledge technique was proposed by Shamir in 1984 [Sha85]. Another important scheme is presented in [FFS88]. [QG90] gives an easy-to-read introduction to zero-knowledge protocols.

In [IS10181-2] zero-knowledge techniques are (falsely) defined as a class of procedures adequate for authentication of single principal or for signature, using a single value of verification authentication information for a set of principals. This property of zero-knowledge techniques makes key distribution much easier than with authentication based on PKCs, where a separate piece of verification authentication information (namely the public key) is needed for each principal. Unfortunately, this nice property is not shared by all zero-knowledge schemes.

Zero-knowledge techniques can be used for purposes of authentication or non-repudiation but usually not for data encryption. Zero-knowledge techniques are most often computationally less complex than PKCs which makes them attractive for user authentication with a smart card. Typical properties of zero-knowledge techniques, according to [FP90], are the following:

- The security level of the verifier depends on the number of iteration cycles of the basic protocol.

- The claimant protects his secret key by giving the verifier at every iteration cycle access only to a randomly selected part of the key.

- One-way functions are used to minimize information flow.

- The verifier does not learn anything during the authentication process that would enable him to impersonate as the claimant later.

Some existing zero-knowledge techniques make key management very simple by completely abolishing the need for user-dependent public keys. The draw-back of these techniques (in comparison with PKCs) is, that they require the secret keys to be generated by a trusted third party (the CA) and they cannot be used for data confidentiality. An example of such an authentication technique is the Fiat-Shamir technique first presented in 1986 [FS87].

A1.6 Physical Mechanisms

In the case of cryptography, the physical mechanisms at the bottom of the hierarchy that are needed to actually perform the cryptographic functions employed can be pieces of software (running on a piece of hardware) or hardware.

Usually the two lowest layers of OSI are implemented in hardware (the Physical layer and the Medium Access Control sublayer of the Data Link layer) whereas the upper layers (from the Logical Link Control sublayer of the Data Link layer up) are implemented in software. However, dedicated hardware can be used at any layer of the OSI model to perform computation-intensive functions, such as cryptography.

The physical mechanisms used have an impact on the over-all security of the system because an implementation of an open system cannot be more secure than the weakest physical mechanism, or the weakest path between these mechanisms, used in the trusted parts of the system. A more detailed analysis of the security of physical mechanisms falls in the area of local systems security and is beyond the scope of this study.

With the transmission speeds offered by current data networks, the efficiency of the physical mechanisms used can become a major issue of systems design. The choice between various physical mechanisms is a trade-off between economy, flexibility, and performance. Therefore, different solutions will be used in different applications. The choice of these mechanisms does usually not affect the external behavior of the system except performance-wise.

However, new portable trusted pieces of software open up new opportunities for the application of secure open systems. The smart card, or its less compact calculator-like version with an integrated keyboard and display, is the only reliable way for strong authentication of the user. There is a very large potential for the use of such devices.

A2 Some Commonly Used Cryptosystems

This work is independent of any individual cryptosystems or other low-level mechanisms. In the preceding chapters we have simply assumed that we have at our disposal symmetric and asymmetric cryptosystems with certain properties. In a constant struggle between cryptographers and cryptanalysts to make and break new cryptosystems existing systems will die and new systems be born. It is important that this work shall not be tied to any currently existing schemes.

However, since we are interested in actual implementations of secure open systems it is important that we be able to specify some current cryptosystems fulfilling the requirements set before. In this chapter we will have a brief look at some cryptosystems which, for the time being, are considered suitable for our purposes.

Here we shall not present the details of these cryptosystems but rather concentrate on their applicability to our purposes. Any good tutorial on cryptology, such as [DH79], [DP84], [Mas88] or [Pat87], will discuss these algorithms in detail.

The schemes presented here are all well known and well established. They have all resisted cryptanalytic attacks for a number of years and are still considered reasonably safe although many of the later cryptosystems have been broken. See e.g. [BO88] for an overview of cryptanalytic attacks against these three cryptosystems as well as some others.

A2.1 DES

Those interested in the history, current use and future prospects of DES are advised to read [SB88] for an overview of the topic.

In 1973 the US National Bureau of Standards (NBS) initiated the development of the Data Encryption Standard (DES) by publishing a solicitation for encryption algorithms for computer data protection. The US National Security Agency (NSA) was requested to assist in the evaluation of proposed encryption schemes.

IBM had started work on developing cryptosystems in the late 60's which had already resulted in commercial applications. In the early 70's the work was continued and it led to a number of patents and products. IBM submitted a cryptographic algorithm to NBS in response to the solicitation and this algorithm was to become DES.

From the very beginning there was a lot of concern about the security of DES. The main grounds for criticism were the following:

Firstly, the 56-bit key-length of DES was not considered adequate by everybody and for all kinds of use. In 1975 Diffie and Hellman proposed a scheme for breaking DES by using massive parallelism and exhaustive search of keys. This would make breaking DES quite costly but possible for organizations such as intelligence services. This scheme is presented in [DH77].

Secondly, the design criteria of the eight substitution tables (called "S-boxes") used in DES were never made public. This led many to believe that there was an intentional trapdoor in DES making it very easy break for NBS and NSA, and very hard for everybody else.

Before accepting the standard, NBS organized two workshops in order to study the security of the proposed algorithm. One was on mathematics and the other on the economic trade-offs of increasing the key-length. Based on the results of these workshops, NBS decided to pass the standard without modifications but it was to be reviewed every five years. The standard was last reviewed in 1988 and passed for another five years. The next revision process will start in early 1992.

Despite extensive cryptanalysis during the past fifteen years and the proposed attacks nobody has to date been able to show a flaw in the design of DES. All of the serious attacks are based on exhaustive search. DES is therefore still considered a reasonably safe algorithm for commercial (but not for military) use. Some of the most recent cryptanalytic studies (see e.g. [BS90]) seem to indicate that the standard 16-round DES still is reasonably safe – in fact much safer than many of its more recent rivals, such as the Japanese FEAL algorithm [SM87].

DES is a symmetric block cipher with a block length of 64 bits and effective key length of 56 bits. The DES standards [ANS81, ANS83] define the basic 64-bit block cipher and four modes of operation for it. These modes are called: Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB) and Output Feedback (OFB). Of these modes, the first one is the basic block cipher and the latter three turn DES into a chained "stream cipher". CBC and OFB are well suited for calculation of cryptosums (e.g. integrity check values) and CFB is good for encrypting individual characters. These modes of operation can also be used with any other block cipher, such as FEAL. Similar definitions now also exist as international standards, [IS8372] defines the modes of operation for 64-bit and [IS10116] for N-bit block ciphers.

As a summary, it can be stated that among the good properties of DES are the following: it is a commonly accepted standard encryption scheme with lots of efficient hardware implementations available, it is computationally reasonably light, it has several standardized modes of operation suited for various purposes, and its safety is still believed to be adequate for most purposes.

Among the shortcomings of DES are the following: the key length is insufficient by today's standards, it is still rumored to contain trapdoors, even though the most recent studies [BS90] don't seem to support this belief, and the initial and final permutations of DES decrease the efficiency of its software implementations.

It is expected that DES will survive yet for a long time in commercial applications, even though in has already been replaced by later, more secure cryptosystems, such as CCEP, in the most critical (e.g. defense-related) applications.

A2.2 RSA

The Rivest-Shamir-Adleman (RSA) algorithm first presented in [RSA78] is the most commonly used and probably most usable Public Key Cryptosystem today. RSA is based on the exponentiation of primes (modulo n) and breaking it is as hard as finding the logarithm (modulo n) of an integer, which is believed to be as hard as factoring a large integer. With key lengths of at least 500 bits RSA is still believed to be reasonably and with about 1000 bits quite safe.

RSA is fairly simple, safe, and adaptive to various kinds of use. Among its shortcomings are its large (and variable) block length and heaviness (RSA is about three orders of magnitude heavier to compute than DES).

As we can see, RSA can easily be used for generating digital signatures. However, the group of possible future algorithms meeting one of the two requirements stated above is presumably much larger than just RSA and we can feel confident that generating digital signatures using a PKC as described here does not limit us into using RSA.

Because breaking RSA is at most as hard as factoring a large integer, some recent advances in the factoring of large integers have made the use of RSA less secure and increased the needed key length. Considering the possibility of a final break-through in factoring, we cannot base everything (authentication, non-repudiation, key distribution etc.) on RSA but have to be ready to quickly adopt new PKCs and authentication schemes if necessary.

A2.3 The Diffie-Hellman Algorithm

The Diffie-Hellman scheme, first proposed in [DH76] as the first published "public-key" algorithm, is still considered one of the best methods for secretly sharing pairwise symmetric keys. The algorithm is based on public "half-keys" and secret values associated with them. From their public half-keys K_A and K_B the communicating parties A and B can determine a pairwise session key K_{AB}, which remains secret from other parties. This key can then be used for mutual authentication and or exchanging secret information (such as a fresh session key).

In fact this algorithm is not a data encryption scheme but a key management scheme where n parties can secretly agree on $n \cdot (n-1)/2$ pairwise encryption keys using only n public "half-keys". Normally $n \cdot (n-1)/2$ pairwise keys would have be created and n-1 keys securely delivered and stored at each site in order to allow for secure communications between any two sites. When using the Diffie-Hellman algorithm only authenticated and integral (not confidential) copies of n-1 public "half-keys" have to be delivered to and a site-specific (locally created) secret number securely stored at each site.

One of the problems associated with the Diffie-Hellman scheme is, that all the parties wishing to communicate with one-another have to use the same parameters values (namely the generator and modulus) of the cryptosystem. Obviously this is not possible on a global scale. Another problem with the Diffie-Hellman algorithm is that, since the master key K_{AB} between A and B is shared by both parties, non-repudiation is not possible without employing a trusted third-party.

Furthermore, with this algorithm each user in the group needs to have an integral copy of every other user's public authentication information in order to establish secure communication with him. This public authentication information either has to be distributed through an authenticated and integral channel or certified by a CA (which requires the use of a public key signature scheme).

These restrictions together make the Diffie-Hellman algorithm only suitable for secure communications in a closed, reasonably sized, fairly stable user group where it is possible to agree on the common parameters and for all users to keep the integral copies of the public authentication information of all the users. Despite its shortcomings, the Diffie-Hellman algorithm is the basis of public key cryptography and even some recent publications, such as [OT89], do propose authentication and key distribution systems based on it.

A3 The Kerberos Authentication Server

Authentication protocols have been widely studied for a long time and it is not our purpose to present a tutorial here. The interested reader is referred to the classical papers [NS78, DS81, NS87, OR87] or any good tutorial on cryptology.

Perhaps the most prominent strong authentication service in wide use today is the Kerberos Authentication Server created in the Athena project at MIT [SNC88]. Kerberos is in everyday use in several major universities (including MIT and Carnegie-Mellon University) and obviously has solved a number of security problems in them. It has also recently been adopted as a standard part of distributions of the DEC Ultrix operating system. Therefore, its significance should not be underestimated. However, it deserves to be pointed out that the approach taken in this work is fundamentally different from that of Kerberos in several respects and is assumed to supersede the currently existing authentication schemes.

In Kerberos authentication is based on symmetric encryption which precludes the stronger service of non-repudiation and leads to the problems of key management described in chapter 4.1.3. Obviously, in an academic environment, repudiation is not considered a serious threat. The way in which key management has been solved in Kerberos is interesting and deserves a brief summary.

To avoid the need for $O(n^2)$ pairwise master keys, each communicating party shares one master key with Kerberos. Kerberos, therefore, is in possession of all the master keys. When using Kerberos, the authentication process proceeds as follows:

1) Client C first asks for a certificate (called "ticket") from Kerberos by sending it the message: "C, TGS"

2) Kerberos authenticates the user by his master key and prevents replay by using time-stamps. Kerberos now passes the ticket for a "Ticket Granting Server" (TGS) to C. This ticket includes a pairwise fresh session key to be used between the C and TGS:
   $T_{C,TGS}$: TGS, C, Addr, TimeStamp, LifeTime, $K_{C,TGS}$
   "$eK_C(K_{C,TGS}, eK_{TGS}(T_{C,TGS}))$"

3) User C now identifies himself to the TGS by presenting his ticket together with an "authenticator" A_C and asks for another ticket to the actual service S he wants to access.
   A_C: C, Addr, TimeStamp
   "S, $eK_{TGS}(T_{C,TGS})$, $eK_{C,TGS}(A_C)$"

4) TGS checks the validity of the ticket and uses the timestamp of the authenticator to detect attempts of replay. TGS now gives C a ticket to S. This ticket includes a pairwise session key to be used between the C and S.
$T_{C,S}$: "$eK_{C,TGS}(eK_S(T_{C,S}), K_{C,S})$"

5) The user can now access the service by presenting his ticket together with an authenticator.
A_C:"$eK_S(T_{C,S})$, $eK_{C,S}(A_C)$"

While this approach works well in limited environments, it still has a number of shortcomings:

- Kerberos is in possession of all the master keys and can therefore impersonate as any of the other parties. Placing this much trust in any one party causes considerable security threats.

- As the number of users and services grows, it will become unfeasible to manage the master keys.

In our approach we only need to trust our own CA and the CAs certified by it. In particular, none of the on-line servers needs to be trusted. Certificates forming an unbroken chain of trust can be received from any untrusted source, such as the Directory. As soon as we know the public keys of our peer, authentication can be done on a bilateral basis between the two communicating end-systems without involving anybody else in the process.

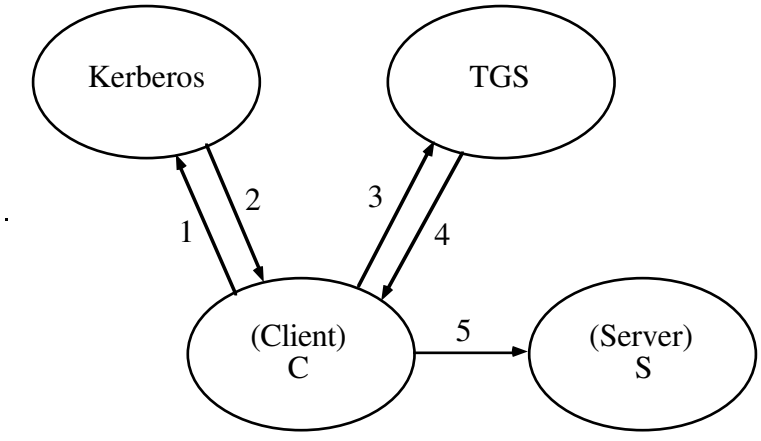

Figure A3-1: The Kerberos Authentication Server.

A4 Security Policy

Security policy is a statement, made with the knowledge and assistance of top management, that says, in very general terms, what is and what is not to be permitted in the area of security during the general operation of the system being secured. That is, a security policy tells "what" rather than "how" is to be done [IS7498-2].

Security policies are conventionally divided into *Identity-Based* and *Rule-Based* policies:

Identity-based security policies are based on privileges or capabilities given to users and/or Access Control Lists associated with data items and other resources. In a rule-based security policy, Security Classes are normally used for determining what is authorized behavior. In identity-based systems, the user traditionally identifies himself by presenting to the system something he knows (e.g. a password). This is often called "need to know" policy.

In rule-based security policies, information and other resources are conventionally divided into *security classes* and the flow of information between (and within) various classes is controlled by a set of rules. There are also rules regarding the security relations between various resources. A classical rule-based security model is the Bell-LaPadula model, based on finite state machines and security classes. The model was first presented in 1973 and a refined version can be found in [BP74].

It should be noted, that besides security classes there is the orthogonal dimension of integrity by which resources can be divided into *integrity classes* first introduced by Biba in [Bib77]. Combining the ideas of Bell, LaPadula and Biba, we can state the following general rules for information flow in a rule-based secure system:

- A subject can only read an object, whose security level is the same or less than that of the subject.

- A subject can only write to an object, whose security level is the same or greater than that of the subject.

- A subject can only read an object, whose integrity level is the same or greater than that of the subject.

- A subject can only write to an object, whose integrity level is the same or less than that of the subject.

A recent paper on rule-based security policies in a distributed environment is [LS90]. Some classical papers on security policies are [Den76] and [Lan81]. An overview of the topic can be found e.g. in [IS7498-2].

In a system everything, including processes, data items and channels, are resources and can be labeled with attributes. Security policies indicate how these attributes are to be used to provide required level of security. The rules are stored in a Security Management Information Base (SMIB).

It is only after an explicit security policy has been stated that security becomes an engineering problem and every organization seriously interested in security should one. The enforcement of this security policy and monitoring of security related events lies in the domain of engineering and means for doing this are discussed in this study.

Security finally boils down into an economic issue. Security is on its optimal level when the sum of total losses caused by security defects and the cost of providing security is at its minimum. The losses caused by security violations are usually very difficult to quantify. However, the optimal security level always lies somewhere in between the two extremes. With no security measures losses can easily grow large. If flawless security is pursued the cost of providing security grows out of all reasonable bounds. The situation is depicted in figure A4-1.

While exactly locating the optimum is impossible, careful risk assessment should be used to approximately locate it. In drawing up a security policy and performing risk assessment also non-technical security measures should be considered. Without appropriate measures e.g. in the areas of Personnel Policy and Physical Security all technical efforts to ensure security are in vain, the security chain always being as strong as its weakest link.

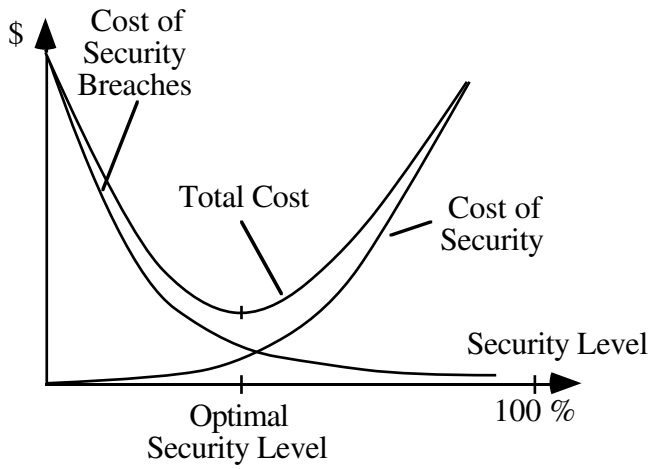

Figure A4-1: Optimal security level.

Bibliography

[Akl83] S. Akl: *Digital Signatures: A Tutorial Survey*, IEEE Computer Magazine, Feb. 1983, pp. 15-24

[AKS90] A. Ahtiainen, J. Keskinen, J. Simolin, K. Tarpila, I. Turunen, *Protocol Software Engineering Tools for Implementation of a General Purpose OSI Stack*, in Computer Networking (L. Csaba et all., editors), Elsevier Science Publishers, BV (North Holland), presented in IFIP conference, Budapest, May 1990

[ANS81] ANSI X3.92, *Data Encryption Algorithm*, American National Standards Institute, New York, 1981

[ANS83] ANSI X3.106, *Data Encryption Algorithm – Modes of Operation*, American National Standards Institute, New York, 1983

[BAN89] M. Burrows, M. Abadi, R. Needham: *A Logic of Authentication*, ACM Operating Systems Review, Vol. 23, No. 5, 1989

[Bar90] R. Barrett, *Security in Wide Area Networks – Some Baseline Requirements*, Computer Fraud & Security Bulletin, Dec. 1990, pp. 14-18

[BDH88] D. Branstad, J. Dorman, R. Housley, *SP4: A Transport Encapsulation Security Protocol*, 1988

[Bel89] S. Bellowin, *Security Problems in the TCP/IP Protocol Suite*, ACM Computer Communication Review, Vol. 19, No. 2, April 1989, pp. 32-48

[BG89] T. Beth, D. Gollmann, *Algorithm Engineering for Public Key Algorithms*, IEEE Journal on Selected Areas in Communications, Vol. 7, No. 4, May 1989, pp. 458-466

[BI89] F. Burg, N. Di Iorio, *Networking of Networks: Interworking According to OSI*, IEEE Journal on Selected Areas in Communications, Vol. 7, No. 7, September 1989, pp. 1131-1142

[Bib77] K. Biba, *Integrity Considerations for Secure Computer Systems*, ESD-TR-76-372, MITRE MTR-3153 NTIS AD A039324, Bedford, Mass., April 1977

[Bir85] A. Birrell, *Secure Communications Using Remote Procedure Calls*, ACM Transactions on Computer Systems, Vol. 3, No. 1, Feb. 1985, pp. 1-14

[BM90] J. Burns, C. Mitchell, *A Security Scheme for Resource Sharing over a Network*, Computers & Security, Vol. 9, No. 1, Feb. 1990, pp. 67-75

[BO88] E. Brickell, A. Odlyzko, *Cryptanalysis: A Survey of Recent Results*, Proceedings of the IEEE, Vol. 76, No. 5, May 1988, pp. 578-593

[BP74] D. Bell, E. LaPadula, *Secure Computer Systems: Mathematical Foundations and Model*, M74-244, Vol. 2, MITRE Corp., Bedford, Mass., Oct. 1984

[BS90] E. Biham, A. Shamir, *Differential Cryptanalysis of DES-Like Cryptosystems*, Proceedings of Crypto'90, June 1990

[BVA90] T. Beth, S. Vanstone, G. Agnew, *What One Should Know about Public Key Algorithms – Today!*, proceedings of SECURICOM 90, Paris, March 1990

[CEN90] *Report on Taxonomy for Security Standardisation*, prepared for the ITSTC by the CEN/CENELEC Security Group, CSecG/49/90, Sep. 1990

[CG90] P.-C. Cheng, V. Gligor, *On the Formal Specification and Verification of a Multiparty Session Protocol*, Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, May 7-9, 1990, pp. 216-233

[CGV90] R. Cocino, M.T. Grillo, F. Vaschetto, *Draft Specification for a Secure Tranport Protocol Class 0*, 4th version, a contribution to COST-225, ISO/IEC JTC1/SC6/WG4, December 1990

[Cha85] D. Chaum, *Security without Identification: Transaction Systems to Make Big Brother Obsolete*, Communications of the ACM, Vol. 28, No. 10, Oct. 1985, pp. 1030-1044

[Cla88] D. Clark, *The Design Philosophy of the DARPA Internet Protocols*, ACM Computer Communication Review, Vol. 18, No. 4, Aug. 1988, pp. 106-114

[Com89] Commission of the European Communities, *EDI in Perspective*, EUR 11883 EN, Brussel – Luxembourg, 1989

[Com90] Commission of the European Communities, *Proposal for a Decision of the Council in the Field of Information Security* (INFOSEC), COM(90) 314 final, Brussel, July 3, 1990

[COS90] Security Mechanisms for Computer Networks, *Extended OSI Security Architecture*, COST-11 Ter Project Report, Volume II, Draft, Oct. 1990

[CP88] J. Crowcroft, K. Paliwoda, *A Multicast Transport Protocol*, ACM Computer Communication Review, Vol. 18, No. 4, Aug. 1988, pp. 247-256

[CPW89] L. Cassel, C. Partridge, J. Westcott, *Network Management Architectures and Protocols: Problems and Approaches*, IEEE Journal on Selected Areas in Communications, Vol. 7, No. 7, September 1989, pp. 1104-1114

[Dav83] D. Davies, *Applying the RSA Digital Signature to Electronic Mail*, IEEE Computer Magazine, Feb. 1983, pp. 55-62

[Dee88] S. Deering, *Multicast Routing in Internetworks and Extended LANs*, ACM Computer Communication Review, Vol. 18, No. 4, Aug. 1988, pp. 55-64

[Den83] D. Denning, *Protecting Public Keys and Signature Keys*, IEEE Computer Magazine, Feb. 1983, pp. 27-35

[DH76] W. Diffie, M. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory,Vol. IT-22, No. 6, Nov. 1976, pp. 644-654

[DH77] W. Diffie, M. Hellman, *Exhaustive Cryptanalysis of the NBS Data Encryption Standard*, Computer, June 1977, pp. 74-78

[DH79] W. Diffie, M. Hellman, *Privacy and Authentication: An Introduction to Cryptography*, Proceedings of the IEEE, Vol. 67, No. 3, Mar. 1979, pp. 397-427

[Dif88] W. Diffie, *The First Ten Years of Public-Key Cryptography*, Proceedings of the IEEE, Vol. 76, No. 5, May 1988, pp. 560-577

[DKK90] F. Dix, M. Kelly, R. Klessig, *Access to a Public Switched Multi-Megabit Data Service Offering*, ACM Computer Communication Review, Vol. 20, No. 3, July 1990, pp. 46-61

[DoD85] *Trusted Computer Systems Evaluation Criteria*, DoD 5200.28-STD, Department of Defense, USA, 1985

[DP84] D. Davies, W. Price, *Security for Computer Networks*, John Wiley & Sons, 1984

[DS81] D. Denning, G. Sacco, *Timestamps in Key Distribution Protocols*, Communications of the ACM, Vol. 24, No. 8, Aug. 1981, pp. 533-536

[ECM88] ECMA, *Security in Open Systems - A Security Framework*, ECMA TR/46, July 1988

[ECM89] ECMA, *Security in Open Systems, Data Elements and Service Definitions*, July 1989

[Far91] Discussions with professsor David Farber in Helsinki, March 1991

[FFS88] U. Feige, A. Fiat, A. Shamir, *Zero-knowledge Proofs of Identity*, Journal of Cryptology, 1, 1988

[FP90] W. Fumy, A. Pfau, *Asymmetric Authentication Schemes for Smart Cards – Dream or reality?*, IFIP TC-11 6th International Conference and Exhibition on Information Security, Espoo, Finland, 1990

[FS87] A. Fiat, A. Shamir, *How to Prove Yourself: Practical Solutions to Identification and Signature Problems*, Proceedings of Crypto'86, Springer, 1987

[Gen90] G. Genilloud, *X.400 MHS: First Steps Towards an EDI Communication Standard*, ACM Computer Communication Review, Vol. 20, No. 2, April 1990, pp. 72-86

[GNY90] L. Gong, R. Needham, R. Yahalom, *Reasoning about Belief in Cryptographic Protocols*, Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, May 7-9, 1990, pp. 234-248

[GOS90] *U.S. Government Open Systems Interconnection Profile* (GOSIP), Draft, Version 2.0, April 1989

[HD89] C. Huitema, A. Doghri, *Defining Faster Transfer Syntaxes for the OSI Presentation Protocol*, ACM Computer Communication Review, Vol. 19, No. 5, Oct. 1989, pp. 44-55

[Hei90] J. Heinänen, *Review of ISO IP*, a contribution to the EC COSINE project, August 13, 1990

[Hei91] J. Heinänen, *Review of Backbone Technologies*, preprint, to be published in the RARE Symposium on High Speed Networking for Research in Europe, Jan. 24, 1991, Brussels

[HKK90] J. Harju, J. Koivisto, J. Kuittinen, J. Lahti, J. Malka, E. Ojanperä, J. Reilly, *C-VOPS Users' Manual*, Technical Research Centre of Finland, Telecommunications laboratory, 1990

[Hou89] R. Housley, *Authentication, Confidentiality, and Integrity Extensions to the XNS Protocol Suite*, ACM Security, Audit & Control Review, Vol. 7, No. 3, Fall 1989, pp. 17-24

[IEE90] Institute of Electrical and Electronic Engineers, Proposed Standard: *DQDB Subnetwork of a Metropolitan Area Network*, IEEE 802.6, P802.6/D14, July 13, 1990

[IM90] C. I'Anson, C. Mitchell, *Security Defects in CCITT Recommendation X.509 – The Directory Authentication Framework*, ACM Computer Communication Review, Vol. 20, No. 2, April 1990, pp. 30-34

[IS7498-1] ISO, Information Processing Systems, *Open Systems Interconnection Reference Model, Part 1: Basic Reference Model*, ISO 7498-1 (CCITT X.200), Geneva 1984

[IS7498-2] ISO, Information Processing Systems, *Open Systems Interconnection Reference Model, Part 2: Security Architecture*, ISO DIS 7498-2, July 19, 1988

[IS7498-3] ISO, Information Processing Systems, *Open Systems Interconnection Reference Model, Part 3: Naming and Addressing*, ISO DIS 7498-3, 1989

[IS7498-4] ISO, Information Processing Systems, *Open Systems Interconnection Reference Model, Part 4: Management Framework*, ISO DIS 7498-4, 1987

[IS7498/A1] ISO, Information Processing Systems, *Open Systems Interconnection, Connectionless Data Transmission*, ISO 7498-1/Add. 1, 1987

[IS7498/A2] ISO, Information Processing Systems, *Open Systems Interconnection, Multipeer Data Transmission*, ISO 7498-1/Add. 2

[IS7498/C1] ISO, Information Processing Systems, *Open Systems Interconnection, Technical Corrigendum 1*, ISO 7498-1/Cor. 1, 1988

[IS7776] *HDLC – Description of the X.25 LAPB-compatible DTE Data Link Procedures*, ISO 7776, 1986

[IS8072/A1] ISO, Information Processing Systems, *Open Systems Interconnection, Transport Service Definition, Addendum 1: Connectionless-mode Transmission*, ISO 8072/Add. 1

[IS8073/A2] ISO, Information Processing Systems, *Open Systems Interconnection, Connection Oriented Transport Protocol Specification, Addendum 2: Class 4 Operation over Connectionless Network Service*, ISO 8073/Add. 2

[IS8073] ISO, Information Processing Systems, *Open Systems Interconnection, Connection Oriented Transport Protocol Specification*, ISO 8073 (CCITT X.224), 1986

[IS8208] ISO, Information Processing Systems, *Open Systems Interconnection, X.25 Packet Level Protocol for Data Terminal, Equipment*, ISO 8208 (CCITT X.25), 1990

[IS8326/A3] ISO, Information Processing Systems, *Open Systems Interconnection, Basic Connection Oriented Session Service Definition, Addendum 3: Connectionless Session Service*, ISO 8326/DAD3, 1988

[IS8348/A1] ISO, Information Processing Systems, *Open Systems Interconnection, Network Service Definition, Addendum 1: Connectionless-mode Transmission*, ISO 8348/AD1, 1987

[IS8348] ISO, Information Processing Systems, *Open Systems Interconnection, Network Service Definition*, ISO 8348 (CCITT X.213), 1987

[IS8372] ISO, Information Technology, Security Techniques, *Modes of Operation for a 64-bit Block Cipher Algorithm*, ISO/IEC, ISO 8372, 1989

[IS8473] ISO, Information Processing Systems, *Open Systems Interconnection, Protocol for Providing the Connectionless-mode Network Service (Internetwork Protocol)*, ISO 8473

[IS8571-1] ISO, Information Processing Systems, *Open Systems Interconnection – File Transfer, Access and Management, Part 1: General Introduction*, ISO 8571-1, Oct. 1988

[IS8571-2] ISO, Information Processing Systems, *Open Systems Interconnection, File Transfer, Access and Management, Part 2: The Virtual Filestore Definition*, ISO 8571-2, Oct. 1988

[IS8571-4] ISO, Information Processing Systems, *Open Systems Interconnection, File Transfer, Access and Management, Part 4: The File Protocol Specification*, ISO 8571-4, Oct. 1988

[IS8602] ISO, Information Processing Systems, *Open Systems Interconnection, Protocol for Providing the Connectionless-mode Transport Service*, ISO 8602, Dec. 1987

[IS8649/A1] ISO, Information Processing Systems, *Open Systems Interconnection, Association Control Service Element, Addendum 1: Authentication*, ISO 8649/DAD1, 1989

[IS8649/A2] ISO, Information Processing Systems, *Open Systems Interconnection, Association Control Service Element, Addendum 2: Connectionless ACSE Service*, ISO 8649/DAD2, 1989

[IS8649/A3] ISO, Information Processing Systems, *Open Systems Interconnection, Association Control Service Element, Addendum 3: A-Context Management Service*, ISO 8649/PDAD3 (working draft), 1990

[IS8649] ISO, Information Processing Systems, *Open Systems Interconnection, Service Definition for the Association Control Service Element (ACSE)*, ISO 8649 (CCITT X.217), 1988

[IS8650/A1] ISO, Information Processing Systems, *Open Systems Interconnection, Peer Entity Authentication During Association Establishment*, ISO 8650/DAD1, 1989

[IS8650/A4] ISO, Information Processing Systems, *Open Systems Interconnection, Application Entity Titles*, ISO 8650/PDAD4 (working draft), 1990

[IS8650] ISO, Information Processing Systems, *Open Systems Interconnection, Protocol Specification for the Association Control Service Element (ACSE)*, ISO 8650 (CCITT X.227), 1988

[IS8802-1] ISO, Information Processing Systems, *Open Systems Interconnection, Local Area Networks, Part 1: Introduction*, ISO 8802-1

[IS8802-2] ISO, Information Processing Systems, *Open Systems Interconnection, Local Area Networks, Part 2: Logical Link Control*, ISO 8802-2

[IS8802-3] ISO, Information Processing Systems, *Open Systems Interconnection, Local Area Networks, Part 3: Carrier Sense Multiple Access with Collision Detection,Access Method and Physical Layer Specifications*, ISO 8802-3, 1989

[IS8802-5] ISO, Information Processing Systems, *Open Systems Interconnection, Local Area Networks, Part 5: Token Ring Access Method and Physical Layer Specifications*, ISO DIS 8802-5, 1990

[IS8807] *LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*, ISO 8807, 1989

[IS8822/A1] ISO, Information Processing Systems, *Open Systems Interconnection, Connection Oriented Presentation Service Definition, Addendum 1: Connectionless-mode Presentation Service*, ISO 8822/DAD1, 1989

[IS8823] ISO, Information Processing Systems, *Open Systems Interconnection, Connection Oriented Presentation Protocol Specification*, ISO 8823 (CCITT X.226), 1988

[IS8824/A1] ISO, Information Processing Systems, *Open Systems Interconnection, Specification of Abstract Syntax Notation One, Addendum 1: ASN.1 Extensions*, ISO 8824/DAD1 (CCITT X.208), 1988

[IS8824] ISO, Information Processing Systems, O*pen Systems Interconnection, Specification of Abstract Syntax Notation One*, ISO 8824 (CCITT X.208), 1987

[IS8825/A1] ISO, Information Processing Systems, *Open Systems Interconnection, Specification of Basic Encoding Rules for ASN.1, Addendum 1: ASN.1 Extensions*, ISO 8825/DAD1 (CCITT X.209), 1988

[IS8825] ISO, Information Processing Systems, *Open Systems Interconnection, Specification of Basic Encoding Rules for ASN.1*, ISO 8825 (CCITT X.209), 1987

[IS8831] ISO, Information Processing Systems, *Open Systems Interconnection, Job Transfer and Manipulation Concepts and Services*, ISO 8831, 1989

[IS8832] ISO, Information Processing Systems, *Open Systems Interconnection, Specification of the Basic Class Protocol for Job Transfer and Manipulation*, ISO 8832, 1989

[IS9040] ISO, Information Processing Systems, *Open Systems Interconnection, Virtual Terminal Service – Basic Class*, ISO 9040, 1988

[IS9041] ISO, Information Processing Systems, *Open Systems Interconnection, Virtual Terminal Protocol – Basic Class*, ISO 9041, 1989

[IS9066-1] ISO, Information Processing Systems, *Open Systems Interconnection, Reliable Transfer, Part 1: Model and Service Definition*, ISO 9066-1 (CCITT X.218), 1989

[IS9066-2] ISO, Information Processing Systems, *Open Systems Interconnection, Reliable Transfer, Part 2: Protocol Specification*, ISO 9066-2 (CCITT X.228), 1989

[IS9072-1] ISO, Information Processing Systems, *Open Systems Interconnection, Remote Operations, Part 1: Model, Notation and Service Definition*, ISO 9072-1 (CCITT X.219), 1989

[IS9072-2] ISO, Information Processing Systems, *Open Systems Interconnection, Remote Operations, Part 2: Protocol Specification*, ISO 9072-2 (CCITT X.229), 1989

[IS9314-1] *Fiber Distributed Data Interface, Part 1: Physical Layer Protocol*, ISO 9314-1, 1989

[IS9314-2] *Fiber Distributed Data Interface, Part 2: Medium Access Control*, ISO 9314-2, 1989

[IS9314-3] *Fiber Distributed Data Interface, Part 3: Physical Layer Medium Dependent*, ISO DIS 9314-3, 1990

[IS9314-4] *Fiber Distributed Data Interface, Part 3: Single Mode Fiber/Physical Layer Medium Dependent*, ISO DP 9314-4, 1990

[IS9314-5] *Fiber Distributed Data Interface, Part 5: Hybrid Ring Control (FDDI II)*, ISO DP 9314-5, 1990

[IS9545/A1] ISO, Information Processing Systems, *Open Systems Interconnection, Application Layer Structure, Addendum 1: Connectionless Operation*, ISO DIS 9545/PDAD1 (working draft), 1989

[IS9545] ISO, Information Processing Systems, *Open Systems Interconnection, Application Layer Structure*, ISO DIS 9545 (CCITT X.200), 1989

[IS9549] ISO, Information Processing Systems, *Open Systems Interconnection, Connectionless Session Protocol*, ISO DIS 9549, 1988

[IS9576] ISO, Information Processing Systems, *Open Systems Interconnection, Connectionless Presentation Protocol Specification*, ISO DIS 9576, 1989

[IS9579-1] ISO, Information Processing Systems, *Open Systems Interconnection, Remote Database Access, Part 1: General Model, Services and Protocol*, ISO DP 9579, 1990

[IS9579-2] ISO, Information Processing Systems, *Open Systems Interconnection, Remote Database Access, Part 2: SQL Specification*, ISO DP 9579, 1990

[IS9594-1] ISO, Information Processing Systems, *Open Systems Interconnection, The Directory, Part 1: Overview of Concepts, Models and Services*, ISO 9594-1 (CCITT X.500), Dec. 1988

[IS9594-8] ISO, Information processing systems, *Open Systems Interconnection, The Directory, Part 8: Authentication framework*, ISO 9594-8 (CCITT X.509), Dec. 1988

[IS9595] ISO, Information Processing Systems, *Open Systems Interconnection, Common Management Information Service Definition, Part 1: Overview*, ISO DIS 9595, 1989

[IS9596] ISO, Information Processing Systems, *Open Systems Interconnection, Common Management Information Protocol (CMIP) Specification*, ISO DIS 9596, 1989

[IS9796] ISO, Information Technology, Security Techniques, *A Signature Algorithm for Short Messages*, ISO DP 9796, 1990

[IS9797] ISO, Information Technology, Security Techniques, *A Data Integrity Mechanism*, ISO DP 9797, 1990

[IS9798-1] ISO, Information Technology, Security Techniques, *Entity Authentication Mechanisms, Part 1: General Model for Entity Authentication Mechanisms*, ISO/IEC DIS 9798-1, 1990

[IS9798-2] ISO, Information Technology, Security Techniques, *Entity Authentication Mechanisms, Part 2: Entity Authentication Using Symmetric Key Techniques*, ISO/IEC DIS 9798-2, 1990

[IS9798-3] ISO, Information Technology, Security Techniques, *Entity Authentication Mechanisms, Part 3: Entity Authentication Using a Public Key Algorithm*, ISO/IEC CD 9798-3, 1990

[IS9799] ISO, Information Processing Systems, *Peer Entity Authentication Using a Public-Key Algorithm with a Two-Way Handshake*, ISO DP 9799, 1988

[IS10000-1] *International Standardized Profiles, Part 1: Taxonomy Framework*, ISO TR 10000-1, 1990

[IS10000-2] *International Standardized Profiles, Part 2: Taxonomy of Profiles*, ISO TR 10000-2, 1990

[IS10035] ISO, Information Processing Systems, *Open Systems Interconnection, Connectionless ACSE Protocol Specification*, ISO DIS 10035, 1989

[IS10040] ISO, Information Processing Systems, *Open Systems Interconnection, Systems Management Overview*, ISO/IEC DP 10040 (CCITT X.701), 1990, June 16, 1990

[IS10116] ISO, Information Technology, Security Techniques, *Modes of Operation for an N-bit Block Cipher Algorithm*, ISO/IEC DIS 10116, 1990

[IS10118-1] ISO, Information Technology, Security Techniques, *Hash Functions for Digital Signatures, Part 1: General*, ISO/IEC CD 10118-1, June 21, 1990

[IS10118-2] ISO, Information Technology, Security Techniques, *Hash Functions for Digital Signatures, Part 2: Hash Functions using a Symmetric Block Cipher Algorithm*, ISO/IEC CD 10118-2, June 21, 1990

[IS10164-7] ISO, Information Processing Systems, *Open Systems Interconnection, Systems Management, Part 7: Security Alarm Reporting Function*, ISO/IEC DIS 10164-7 (CCITT X.736), Soeul, May 21-22, 1990

[IS10164-8] ISO, Information Processing Systems, *Open Systems Interconnection, Systems Management, Part 8: Security Audit Trail Function*, ISO/IEC DIS 10164-8 (CCITT X.740), Soeul, June 1990

[IS10164-9] ISO, Information Processing Systems, *Open Systems Interconnection, Systems Management, Part 9: Objects and Attributes for Access Control*, ISO/IEC DIS 10164-9 (CCITT X.741), Soeul, June 1990

[IS10165-1] ISO, Information Processing Systems, *Open Systems Interconnection, Structure of Management Information, Part 1: Management Information Model*, ISO/IEC DIS 10165-1 (CCITT X.720), Paris, Jan. 1990

[IS10167] *Guidelines for the Application of Estelle, LOTOS, and SDL*, ISO DTR 8807, 1990

[IS10181-1] ISO, Information Technology, *OSI Security Model, Part 1: Security Framework*, ISO DP 10181-1

[IS10181-2] ISO, Information Technology, *OSI Security Model, Part 2: Authentication Framework*, ISO DP 10181-1

[ISO90a] ISO, Information Processing Systems, *Proposed Draft for End System to End System Security Protocol*, 2nd version, ISO/IEC JTC1/SC6, 27 July, 1990

[ISO90b] ISO, Information Processing Systems, *Appendix B to UK Proposal for Network Layer End System to End System Security Protocol*, ISO/IEC JTC1/SC6, July 27, 1990

[ISO90c] ISO, Information Processing Systems, *OSI Upper Layers Security Model*, fourth working draft, ISO/IEC JTC1/SC21 N5447, ISO/IEC JTC1/SC21/WG6 and CCITT Question 19/VII Collaborative ad hoc Meeting on Security in Berlin, Oct. 30, 1990

[ISO90d] ISO, Information Processing Systems, *Working Draft Integrity Framework*, ISO/IEC JTC1/SC21 N5047, July 1990

[ISO90e] ISO, Information Processing Systems, *Working Draft Non-Repudiation Framework*, ISO/IEC JTC1/SC21 N5046, July, 1990

[ISO90f] D. Kim, Y. Kim, H. Shin, H. Choi, T. Park, *Proposal of a Common Security Service Element in the OSI Application Layer*, ISO/IEC JTC1/SC21 N5002 Attachment, Sep. 1990

[ITS90] *Information Technology Security Evaluation Criteria (ITSEC), Harmonised Criteria of France – Germany – the Netherlands – the United Kingdom*, Draft, May 2, 1990

[Kar86] A. Karila, *Portable Protocol Development and Run-Time Environment*, Licentiate's thesis, Helsinki University of Technology, Apr. 1986

[Kar87] A. Karila, *C-VOPS – A Portable Environment for Protocol Development and Implementation*, Proceedings of IFIP TC6 International Conference on Information Network and Data Communication, Sweden, May 11-14, 1986, North Holland, 1987, pp. 19-34

[Kat90] Dave Katz, *The Use of Connectionless Network Layer Protocols over FDDI Networks*, ACM Computer Communication Review, Vol. 20, No. 3, July 1990, pp. 32-45

[Ken89] S. Kent, *Comments on "Security Problems in the TCP/IP Protocol Suite"*, ACM Computer Communication Review, Vol. 19, No. 3, July 1989, pp. 10-19

[KHA90] A. Karila, J. Harju, I. Airaksinen, M. Sievänen, *Inserting Security Mechanisms into OSI Layers – a Case Study*, Lappeenranta University of Technology, 1990

[KLP83] A. Karila, K. Lång, P. Pulli, Tietoliikennesovittimen toiminta ja rakenne, (*Functional Specification and Structure of a Transport Layer Gateway*, in Finnish), Technical Research Centre of Finland, Research Notes 252, 1983, 94 p.

[KM91] J. Koivisto, J. Malka, *DVOPS – An Object-Oriented Approach to Distributed Computation*, to be presented in the Usenix'91 conference, Technical Research Centre of Finland, January 1991

[Koi87] J. Koivisto, Sovelluskerroksen tietoyksiköiden automaattinen käsittely standardikuvauskielen perusteella, (*Automatic Handling of Application Layer Data Units Based on a Standard Description Language*, in Finnish), Master's Thesis, Helsinki University of Technology, Department of Technical Physics, 1987

[Lan81] C. Landwehr, *Formal Models for Computer Security*, ACM Computing Surveys, Vol. 13, No. 3, Sept. 1981, pp. 247-278

[LLH89] C.-S.Laih, J.-Y. Lee, L. Harn, Y.-K. Su, *Linearly Shift Knapsack Public-Key Cryptosystem*, IEEE Journal on Selected Areas in Communications, Vol. 7, No. 4, May 1989, pp. 534-547

[LS90] W.-P. Lu, M. Sundareshan, *A Model for Multilevel Security in Computer Networks*, IEEE Transactions on Software Engineering, Vol. 16, No. 6, June 1990, pp. 647-659

[LT90] L. Landweber, M. Tasman, *An ISO TP4-TP0 Gateway*, ACM Computer Communication Review, Vol. 20, No. 2, April 1990, pp. 16-21

[Mas88] J. Massey, *An Introduction to Contemporary Cryptology*, Proceedings of the IEEE, Vol. 76, No. 5, May 1988, pp. 533-549

[Mer89] R. Merkle, *One Way Hash Functions and DES*, Proceedings of Crypto'89, Santa Barbara, October, 1989

[MH78] R. Merkle, M. Hellman, *Hiding Information and Signatures in Trapdoor Knapsack*, IEEE Transactions on Information Theory, Vol. IT-24, Sep. 1978, pp. 525-530

[Mit90] C. Mitchell, *OSI and X.400 Security*, Telecommunications, May 1990, pp. 49-54

[MM83] R. DeMillo, M. Merritt, *Protocols for Data Security*, IEEE Computer Magazine, Feb. 1983, pp. 39-51

[MOI90] S. Miyaguchi, K. Ohta, M. Iwata, *128-bit hash function (N-Hash)*, proceedings of SECURICOM90, Paris, March 1990

[Moo88] J. Moore, *Protocol Failures in Cryptosystems*, Proceedings of the IEEE, Vol. 76, No. 5, May 1988, pp. 594-602

[MT79] R. Morris, K. Thompson, *Password Security: A Case History*, Communications of the ACM, Vol. 22, Nov. 1979, pp. 594-597

[MWR89] C. Mitchell, M. Walker, D. Rush, *CCITT/ISO Standards for Secure Message Handling*, IEEE Journal on Selected Areas in Communications, Vol. 7, No. 4, May 1989, pp. 517-524

[NCS87] *Trusted Network Interpretation of the Trusted Computer Systems Evaluation Criteria*, NCSG-TG-005, Version 1, National Computer Security Center, USA July 31, 1987

[Nel88] R. Nelson, *SDNS Services and Architecture*, 1988

[Nes90] D. Nessett: *A Critique of the Burrows, Abadi and Needham Logic*, ACM Operating Systems Review, 1990

[Nok90] *How to Use CASN Compiler for Implementation of a Virtual Task in CVOPS*, Ver. 1.0, Technical Report, Nokia Research Centre, 1990

[NS78] R. Needham, R. Schroeder, *Using Encryption for Authentication in Large Networks of Computers*, Communications of the ACM, Vol. 21, No. 12, Dec. 1978, pp. 993-999

[NS87] R. Needham, R. Schroeder, *Authentication Revisited*, ACM Operating Systems Review, Vol. 21, No. 1, Jan. 1987, p. 7

[NS89] K. Nakao, K. Suzuki, *Proposal on a Secure Communications Service Element (SCSE) in the OSI Application Layer*, IEEE Journal on Selected Areas in Communications, Vol. 7, No. 4, May 1989, pp. 505-516

[OR87] D. Otway, O. Rees, *Efficient and Timely Mutual Authentication*, ACM Operating Systems Review, Vol. 21, No. 1, Jan. 1987, pp. 8-10

[OT89] E. Okamoto, K. Tanaka, *Key Distribution System Based on Identification Information*, IEEE Journal on Selected Areas in Communications, Vol. 7, No. 4, May 1989, pp. 481-485

[Par90] G. Parulkar, *The Next Generation of Internetworking*, ACM Computer Communication Review, Vol. 20, No. 1, Jan. 1990, pp. 18-43

[Pat87] W. Patterson: *Mathematical Cryptology for Computer Scientists and Mathematicians*, Rowman & Littlefield, 1987

[PK79] G. Popek, C. Kline, *Encryption and Secure Computer Networks*, ACM Computing Surveys, Vol. 11, No. 4, Dec. 1979, pp. 331-356

[PK90] D. Piscitello, M. Kramer, *Internetworking Using Switched Multimegabit Data Service in TCP/IP Environments*, ACM Computer Communication Review, Vol. 20, No. 3, July 1990, pp. 62-71

[PSI90] PSI Inc., *ISODE 6.0 Announcement*, Jan. 1990

[QG90] J.-J. Quisquater, L. Guillou, *How to Explain Zero-Knowledge Protocols to Your Children*, 1990

[RFC81a] *Internet Protocol*, RFC 791, Sep. 1981

[RFC81b] *Transmission Control Protocol*, RFC 793, Sep. 1981

[RFC89a] J. Linn, *Privacy Enhancement for Internet Electronic Mail: Part I: Message Encipherment and Authentication Procedures*, RFC 1113, 1989

[RFC89b] S. Kent, J. Linn, *Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management*, RFC 1114, 1989

[RFC89c] J. Linn, *Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes and Identifiers*, RFC 1115, 1989

[RHF90] F. Ross, J. Hamstra, R. Fink, *FDDI – A LAN Among MANs*, ACM Computer Communication Review, Vol. 20, No. 3, July 1990, pp. 16-31

[Riv90] R. Rivest, *The MD4 Message Digest Algorithm* (Version 2/17/90, Revised), ISO/IEC JTC1/SC 27/WG20.2 N193, April 4, 1990

[Ros90] M. Rose, *The Open Book, OSI – a Practical Perspective*, Prentice Hall, 1990

[RSA78] R. Rivest, A. Shamir, L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM, Vol. 21, No. 2, Feb. 1978, pp. 120-126

[Rue90] R. Rueppel, *A Formal Approach to Security Architectures*, preprint of paper to be presented in Eurocrypt'91, April 8 – 11, 1991, Brighton, England, dated Dec. 17, 1990

[Rue91] Lecture by and discussion with Rainer Rueppel at Helsinki University, March 18, 1991

[SDN89a] SDNS, *Secure Data Network Systems, Security Protocol 3 (SP3)*, Specification SDN.301, Revision 1.4, Feb. 28, 1989

[SDN89b] SDNS, *Secure Data Network Systems, Security Protocol 4 (SP4)*, Specification SDN.401, Revision 1.3, May 2, 1989

[SEG90] Security EDIFACT Group, *Security Framework for EDIFACT*, MD4.B, Document 1.19, v. 1.0, (draft), 7 June, 1990

[Sha49] C. Shannon, *Communication Theory of Secrecy Systems*, Bell System Technical Journal, Vol. 28, Oct. 1949, pp. 656-715

[Sha85] A. Shamir, *Identity-Based Cryptosystem and Signature Scheme*, Advances in Cryptology: Proceedings of Crypto'84, Springer, Berlin, 1985, pp. 47-53

[Sim79] G. Simmons, *Symmetric and Asymmetric Encryption*, ACM Computing Surveys, Vol. 11, No. 4, Dec. 1979, pp. 305-330

[Sim84] G. Simmons, *Authentication Theory/Coding Theory*, Advances in Cryptology, Proceedings of Crypto '84, Springer, New York, 1985, pp. 411-431

[Sim88] G. Simmons, *A Survey of Information Authentication*, Proceedings of the IEEE, Vol. 76, No. 5, May 1988, pp. 603-620

[Sim89] G. Simmons, *A Protocol to Provide Verifiable Proof of Identity and Unforgeable Transaction Receipts*, IEEE Journal on Selected Areas in Communications, Vol. 7, No. 4, May 1989, pp. 435-447

[SM87] A. Shimizu, S. Miyaguchi, *Fast Data Encipherment Algorithm FEAL*, Proceedings of Eurocrypt'87, April 1987

[SNC88] J. Steiner, C. Neuman, J. Schiller, *Kerberos: An Authentication Service for Open Network Systems*, Project Athena, MIT, 1988

[TED89] The TEDIS – EDI Security Workshop, *Security in a Multi-Owner System*, Brussels, June 20-21, 1989

[TED90a] *TEDIS Programme 1988-1989*, Activity Report, Brussels, July 25, 1990

[TED90b] *Digital Signatures in EDIFACT*, a TEDIS programme report prepared by CRYPTOMATHIC A/S, final version, Nov. 29, 1990

[TTY88] Kari Lång (editor), Tietotekniikan kehittämisohjelma FINPRIT: Tietotekniikan yhdentäminen, *final report (concise version) of the FINPRIT research program* (in Finnish), TEKES, Helsinki, 1988

[VK83] V. Voydock, S. Kent, *Security Mechanisms in High-Level Network Protocols*, Computing Surveys, Vol. 15, June 1983, pp. 135-171

[Wal89] S. Walker, *Network Security: The Parts of the Sum*, Proceedings of the 1989 IEEE Computer Society Symposium on Security and Privacy, Oakland, 1989, pp. 2-9

[Woo87] C. Wood & al., *Computer Security: A Comprehensive Controls Checklist*, Wiley, New York, 1987

[Woo90] C. Wood, *Principles of Secure Information Systems Design*, Computers & Security, Vol. 9, No. 1, Feb. 1990, pp. 13-24

[X.21] *Interface between data terminal equipment (DTE) and data circuit-terminating-equipment (DCE) for synchronous operation on public data networks*, CCITT Recommendation X.21, Blue Book Vol. VIII, Fascicle VIII.2, Geneva, 1989

[X.25] *Interface between data terminal equipment (DTE) and data circuit-terminating-equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuit*, CCITT Recommendation X.25 (see also ISO 7776 and ISO 8208), Blue Book Vol. VIII, Fascicle VIII.2, Geneva, 1989

[X.200] *Reference Model of Open Systems Interconnection for CCITT Applications*, CCITT Recommendation X.200 (ISO 7498), Blue Book Vol. VIII, Fascicle VIII.4, Geneva, 1989

[X.224] *Transport Protocol Specification*, CCITT Recommendation X.224 (ISO 8073), Blue Book Vol. VIII, Fascicle VIII.5, Geneva, 1989

[X.208] *Specification of Abstract Syntax Notation One (ASN.1)*, CCITT Recommendation X.208 (ISO 8824, ISO 8824/AD1), Blue Book Vol. VIII, Fascicle VIII.4, Geneva, 1989

[X.209] *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*, CCITT Recommendation X.208 (ISO 8825, ISO 8825/AD1), Blue Book Vol. VIII, Fascicle VIII.4, Geneva, 1989

[X.210] *Layer Service Definition Conventions*, CCITT Recommendation X.210 (ISO TR 8509), Blue Book Vol. VIII, Fascicle VIII.4, Geneva, 1989

[X.213] *Network Service Definition for Open Systems Interconnection for CCITT Applications*, CCITT Recommendation X.213 (ISO 8348, ISO 8348/AD2, ISO 8348/AD3), Blue Book Vol. VIII, Fascicle VIII.4, Geneva, 1989

[X.217] *Association Control Service Definition for Open Systems Interconnection for CCITT Applications*, CCITT Recommendation X.217 (ISO 8649), Blue Book Vol. VIII, Fascicle VIII.4, Geneva, 1989

[X.218] *Reliable Transfer: Model and Service Definition*, CCITT Recommendation X.218 (ISO 9066-1), Blue Book Vol. VIII, Fascicle VIII.4, Geneva, 1989

[X.219] *Remote Operations: Model, Notation and Service Definition*, CCITT Recommendation X.219 (ISO 9072-1), Blue Book Vol. VIII, Fascicle VIII.4, Geneva, 1989

[X.224] *Transport Protocol Specification*, CCITT Recommendation X.224 (ISO 8073), Blue Book Vol. VIII, Fascicle VIII.5, Geneva, 1989

[X.227] *Protocol Specification for the Association Control Service Element*, CCITT Recommendation X.227 (ISO 8650), Blue Book Vol. VIII, Fascicle VIII.5, Geneva, 1989

[X.228] *Protocol Specification for the Reliable Transfer Service Element*, CCITT Recommendation X.228 (ISO 9066-2), Blue Book Vol. VIII, Fascicle VIII.5, Geneva, 1989

[X.229] *Protocol Specification for the Remote Operations Service Element*, CCITT Recommendation X.229 (ISO 9072-2), Blue Book Vol. VIII, Fascicle VIII.5, Geneva, 1989

[X.400] *Message Handling, Part 1: System and Service Overview*, CCITT Recommendation X.400, Blue Book Vol. VIII, Fascicle VIII.7, Geneva, 1989

[X.500] *The Directory – Overview of Concepts, Models, and Services*, CCITT Recommendation X.500 (Melbourne 1988, ISO 9594-1), Blue Book Vol. VIII, Fascicle VIII.8, Geneva, 1989

[X.509] *The Directory, Part 8: Authentication Framework*, CCITT Recommendation X.509 (Melbourne 1988, ISO 9594-1), Blue Book Vol. VIII, Fascicle VIII.8, Geneva, 1989

[Z.100] *Functional Specification and Description Language (SDL)*, CCITT Recommendations Z.100 and Z.110, Blue Book Vol. X, Fascicle X.1, Geneva, 1989

[ZE90] R. Zamparo, G. Endersz, *Architecture and Testbed Realization of Security Services in an OSI Communication Environment*, preprint of presentation at ICCC in Nov. 1990 at New Delhi, Sep. 4, 1990

Glossary

ACSE	Association Control Service Element
ANSI	American National Standards Institute
APDU	Application Protocol Data Unit
ARQ	Automatic Repeat Request
ASE	Application Service Element
ASN.1	Abstract Syntax Notation 1
ATM	Asynchronous Transfer Mode
BER	Basic Encoding Rules
CA	Certification Authority
CBC	Cipher Block Chaining
CCITT	Comité Consultatif International de Telephonique et Telegraphique (International Consultative Committee of Telephony and Telegraphy)
CEP	Connection End-Point
CEPI	Connection End-Point Identifier
CFB	Cipher Feed-Back
CLNP	Connectionless Network Protocol
CLNS	Connectionless Network Service
CONS	Connection-Oriented Network Service
C-VOPS	C-language Virtual Operating System
DES	Data Encryption Standard
DIS	Draft International Standard
DoD	Department of Defense
DP	Draft Proposal

DSA	Directory Server Agent
DUA	Directory User Agent
D-VOPS	Distributed Virtual Operating System
ECB	Electronic Code Bood
EDI	Electronic Data Interchange
EDP	Electronic Data Processing
EESP	End-to-End Security Protocol
EESP	End-to-End Security Protocol
EFSA	Extended Finite State Automaton
EFT	Electronic Funds Transfer
FDDI	Fiber Distributed Data Interface
FEAL	Fast Encryption Algorithm
FTAM	File Transfer, Access and Management
FTP	File Transfer Protocol
Gb	Gigabit (1,000,000,000 bits)
GOSIP	Government OSI Profile
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IS	International Standard
ISDN	Integrated Services Digital Network
ISO	International Organization of Standardization
ISODE	ISO Development Environment
JTM	Job Transfer and Manipulation
kb	Kilobit (1,000 bits)

LAN	Local Area Network
LLC	Logical Link Control
MAC	Medium Access Control
MAN	Metropolitan Area Network
Mb	Megabit (1,000,000 bits)
MHS	Message Handling System
MOTIS	Message-Oriented Text Interchange System
MS	Message Store
MTA	Message Transfer Agent
MTS	Message Transfer System
NBS	National Bureau of Standards (now NIST)
NCR	Network Connection Request (PDU)
(N)CR	(N) Connection Request (PDU)
N-CR	Network Connect Request (service primitive)
(N)-CR	(N) Connect Request (service primitive)
NIST	National Institute of Standardization (formerly NBS)
NSA	National Security Agency
NPDU	Network Protocol Data Unit
(N)PDU	(N) Protocol Data Unit
NSAP	Network Service Access Point
(N)SAP	(N) Service Access Point
NSDU	Network Service Data Unit
(N)SDU	(N) Service Data Unit
OFB	Output Feed-Back

OSI	Open Systems Interconnection
P1	Protocol 1 (MTA-to-MTA protocol in X.400 MHS)
P2	Protocol 2 (UA-to-UA protocol in X.400 MHS)
PBX	Private Branch Exchange
PDN	Public Data Network
PDU	Protocol Data Unit
PIN	Personal Identity Number
PKC	Public Key Cryptosystem
PPDU	Presentation Protocol Data Unit
PSDU	Presentation Service Data Unit
PSTN	Public Switched Telephone Network
RDA	Remote Database Access
RFC	Request for Comment
ROS	Remote Operations
ROSE	Remote Operations Service Element
RSA	Rivest–Shamir–Adelman (a public key cryptosystem)
RTS	Reliable Transfer
SASE	Specific Application Service Element
SCSE	Secure Communications Service Element
SDU	Service Data Unit
SMIB	Security Management Information Base
SP3	Network Layer Security Protocol
SP4	Transport Layer Security Protocol
TCP	Transmission Control Protocol

TP0	Class 0 Transport Protocol
TP4	Class 4 Transport Protocol
TPDU	Transport Protocol Data Unit
UA	User Agent
VAN	Value Added Network
VANS	Value Added Network Service
VLSI	Very Large Scale Integration
VOPS	Virtual Operating System
VT	Virtual Terminal
WAN	Wide Area Network
XNS	Xerox Network Systems