X/Open Guide

Distributed Security Framework

X/Open Company Ltd.

X/Open Guide

Distributed Security Framework

ISBN: 1-85912-071-7
X/Open Document Number: G410

Published by X/Open Company Ltd., U.K.

Any comments relating to the material contained in this document may be submitted to X/Open at:

X/Open Company Limited
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

XoSpecs@xopen.co.uk

Contents

List of Figures

List of Tables

Preface

X/Open

X/Open is an independent, worldwide, open systems organisation supported by most of the world's largest information systems suppliers, user organisations and software companies. Its mission is to bring to users greater value from computing, through the practical implementation of open systems.

X/Open's strategy for achieving this goal is to combine existing and emerging standards into a comprehensive, integrated, high-value and usable open system environment, called the Common Applications Environment (CAE). This environment covers the standards, above the hardware level, that are needed to support open systems. It provides for portability and interoperability of applications, and so protects investment in existing software while enabling additions and enhancements. It also allows users to move between systems with a minimum of retraining.

X/Open defines this CAE in a set of specifications which include an evolving portfolio of application programming interfaces (APIs) which significantly enhance portability of application programs at the source code level, along with definitions of and references to protocols and protocol profiles which significantly enhance the interoperability of applications and systems.

The X/Open CAE is implemented in real products and recognised by a distinctive trade mark — the X/Open brand — that is licensed by X/Open and may be used on products which have demonstrated their conformance.

X/Open Technical Publications

X/Open publishes a wide range of technical literature, the main part of which is focussed on specification development, but which also includes Guides, Snapshots, Technical Studies, Branding/Testing documents, industry surveys, and business titles.

There are two types of X/Open specification:

- *CAE Specifications*

  CAE (Common Applications Environment) specifications are the stable specifications that form the basis for X/Open-branded products. These specifications are intended to be used widely within the industry for product development and procurement purposes.

  Anyone developing products that implement an X/Open CAE specification can enjoy the benefits of a single, widely supported standard. In addition, they can demonstrate compliance with the majority of X/Open CAE specifications once these specifications are referenced in an X/Open component or profile definition and included in the X/Open branding programme.

  CAE specifications are published as soon as they are developed, not published to coincide with the launch of a particular X/Open brand. By making its specifications available in this way, X/Open makes it possible for conformant products to be developed as soon as is practicable, so enhancing the value of the X/Open brand as a procurement aid to users.

- *Preliminary Specifications*

  These specifications, which often address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations, are released in a controlled manner for the purpose of validation through implementation of products. A Preliminary specification is not a draft specification. In fact, it is as stable as X/Open can make it, and on publication has gone through the same rigorous X/Open development and review procedures as a CAE specification.

  Preliminary specifications are analogous to the *trial-use* standards issued by formal standards organisations, and product development teams are encouraged to develop products on the basis of them. However, because of the nature of the technology that a Preliminary specification is addressing, it may be untried in multiple independent implementations, and may therefore change before being published as a CAE specification. There is always the intent to progress to a corresponding CAE specification, but the ability to do so depends on consensus among X/Open members. In all cases, any resulting CAE specification is made as upwards-compatible as possible. However, complete upwards-compatibility from the Preliminary to the CAE specification cannot be guaranteed.

In addition, X/Open publishes:

- *Guides*

  These provide information that X/Open believes is useful in the evaluation, procurement, development or management of open systems, particularly those that are X/Open-compliant. X/Open Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming X/Open conformance.

- *Technical Studies*

  X/Open Technical Studies present results of analyses performed by X/Open on subjects of interest in areas relevant to X/Open's Technical Programme. They are intended to communicate the findings to the outside world and, where appropriate, stimulate discussion and actions by other bodies and the industry in general.

- *Snapshots*

  These provide a mechanism for X/Open to disseminate information on its current direction and thinking, in advance of possible development of a Specification, Guide or Technical Study. The intention is to stimulate industry debate and prototyping, and solicit feedback. A Snapshot represents the interim results of an X/Open technical activity. Although at the time of its publication, there may be an intention to progress the activity towards publication of a Specification, Guide or Technical Study, X/Open is a consensus organisation, and makes no commitment regarding future development and further publication. Similarly, a Snapshot does not represent any commitment by X/Open members to develop any specific products.

Versions and Issues of Specifications

As with all *live* documents, CAE Specifications require revision, in this case as the subject technology develops and to align with emerging associated international standards. X/Open makes a distinction between revised specifications which are fully backward compatible and those which are not:

- a new *Version* indicates that this publication includes all the same (unchanged) definitive information from the previous publication of that title, but also includes extensions or additional information. As such, it *replaces* the previous publication.

- a new *Issue* does include changes to the definitive information contained in the previous publication of that title (and may also include extensions or additional information). As such, X/Open maintains *both* the previous and new issue as current publications.

Corrigenda

Most X/Open publications deal with technology at the leading edge of open systems development. Feedback from implementation experience gained from using these publications occasionally uncovers errors or inconsistencies. Significant errors or recommended solutions to reported problems are communicated by means of Corrigenda.

The reader of this document is advised to check periodically if any Corrigenda apply to this publication. This may be done either by email to the X/Open info-server or by checking the Corrigenda list in the latest X/Open Publications Price List.

To request Corrigenda information by email, send a message to info-server@xopen.co.uk with the following in the Subject line:

```
request corrigenda; topic index
```

This will return the index of publications for which Corrigenda exist.

This Document

This document is a Guide (see above). This document describes the X/Open Distributed Security Framework (XDSF). It identifies information system security services required to meet the needs of distributed information systems. It provides guidance on the specification and integration of these services.

This document is structured as follows:

- Chapter 1 explains the importance of security in information systems; readers who are knowledgeable in this area may skip this part. Everyone should read Section 1.4 on page 4 which explains the purpose and scope of this document.
- Chapter 2 introduces the concepts of the security framework, and refers to the sections that explain each concept in more detail.
- Chapter 3 introduces the top-level IT system security concepts, terms and models used within the Framework.
- Chapter 4 introduces the concept of a system security architecture and outlines the layering of security services within such an architecture. It also describes the characteristics of the interfaces to security services. These are particularly important to those designing APIs to security services.
- Chapter 5 describes the basic security services.
- Chapter 6 describes the domain interaction services, the highest of the security service layers, together with a set of supporting services.
- Chapter 7 provides a top-level analysis of the security obligations placed on major functional areas with examples.
- Appendix A outlines the range of security threats to and vulnerabilities within information systems. The analysis of the security threats, vulnerabilities and methods of attack includes all of the areas that will eventually have to be addressed by the framework. This appendix provides background information, which is particularly useful for those who are not security experts.

- Appendix B discusses the quality of availability of a system.
- Appendix C presents interpretations of some common system types in terms of the framework.
- Appendix D outlines an example set of criteria for assessing the relative merits of standardising an interface and the relative merits of particular candidates.
- Appendix E provides simplified examples of the approach needed to integrate security into the specification of APIs.
- Appendix F is to help those developing X/Open specifications, which may make use of security services, incorporate the appropriate security clauses into their specifications.
- A comprehensive glossary and index are also provided.

Intended Audience

This document is intended for programmers who are concerned with:

- future specifications of security services developed by X/Open
- other specifications developed by X/Open
- application development.

Readers are expected to have some knowledge of security terminology. If you have not previously read any documents concerning security in information systems, you should read the **Procurement Guide** (see **Referenced Documents** on page xvii) before reading this guide.

Typographical Conventions

The following typographical conventions are used throughout this document:

- *Italic* strings are used for emphasis or to identify the first instance of a word requiring definition.

Trade Marks

UNIX® is a registered trade mark in the United States and other countries, licensed exclusively through X/Open Company Limited.

X/Open™ and the ‘‘X’’ device are trade marks of X/Open Company Limited.

Acknowledgements

X/Open gratefully acknowledges that this document is the result of a cooperative effort and exchange of ideas between the X/Open Security Working Group and the IEEE POSIX Security Working Group. This cooperation has the objective of ensuring that the technical aspects of the security frameworks being produced by both groups are consistent and as far as possible identical.

Referenced Documents

The following documents are referenced in this guide.

CESG Memo
CESG Memorandum No.1 Issue 1.2 Oct 1992, Glossary of Security Terminology.

ECMA
Authentication and Privilege Attribute Security Application with Related Key Distribution Functions, Parts 1, 2 and 3.

ECMA-138
ECMA-138 Dec 1989, Security in Open Systems — Data Elements and Service Definitions.

ECMA TR/46
Security in Open Systems, A Security Framework, July 1988, European Computer Manufacturers Association.

Federal Criteria
Federal Criteria Version 1.0 Dec 1992, Federal Criteria for Information Technology Security.

ISO/IEC 7498-2
ISO/IEC 7498-2: 1989, Information Processing Systems — Open Systems Interconnection — Basic Reference Model — Part 2: Security Architecture.

ISO/IEC 10181
ISO/IEC 10181, Information Technology — Open Systems Interconnection — Security Frameworks in Open Systems —

10181-1: Part 1: Security Frameworks Overview
10181-2: Part 2: Authentication Framework
10181-3: Part 3: Access Control
10181-4: Part 4: Non-repudiation Framework
10181-5: Part 5: Integrity Framework
10181-6: Part 6: Confidentiality Framework
10181-7: Part 7: Security Audit Framework

ISO/IEC 10745
ISO/IEC DIS 10745, Upper Layers Security Model. May 1992.

ITSEC
Information Technology Security Evaluation Criteria, Provisional Harmonised Criteria, June 1991, Version 1.2, published by the Commission of the European Communities.

POSIX.0
IEEE Std 1003.0/D15, June 1992, Draft Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 0.

POSIX.1e
IEEE Draft Std P1003.1e/D14, Protection, Audit and Control Interfaces, MAR94.

RFC 1510
Internet Proposed Standard, The Kerberos Network Authentication System, John Kohl, B.Clifford Neuman, issue 5.2, 21 April 1993.

TCSEC
Trusted Computer System Evaluation Criteria. U.S. Department of Defense (DOD), 1985

DOD 5200.28-STD, National Computer Security Center, Fort Meade, Md. (also known as the Orange Book).

SESAME
: SESAME — An Introduction, SEP94, SESAME Project. Also refer to the paper to be published in the proceedings of FEB95 Internet Society Symposium on Network and Distributed Systems Security, SESAME V2 — Public Key and Authorisation Extensions to Kerberos.

TSIX(RE)
: Trusted System Interoperability Group (TSIG) Secure Information eXchange for Restricted Environment).

X.509
: ISO/IEC 9594-8: 1990 Information Technology — Open Systems Interconnection — The Directory — Part 8: Authentication Framework, together with:

  Technical Corrigendum 1: 1991 to ISO/IEC 9594-8: 1990.

The following X/Open documents are referenced in this specification:

Auditing and Authentication
: X/Open Snapshot, October 1990, Security Interface Specifications: Auditing and Authentication (S020 or XO/SNAP/90/020).

Base GSS-API
: X/Open Preliminary Specification, January 1994, Generic Security Service API (GSS-API) Base (ISBN: 1-85912-025-3, P308).

GSS-API Extensions
: X/Open Snapshot, January 1994, Generic Security Service API (GSS-API) Security Attribute and Delegation Extensions (ISBN: 1-85912-261-1, S307).

DCE RPC
: X/Open CAE Specification, August 1994, X/Open DCE: Remote Procedure Call (ISBN: 1-85912-041-5, C309).

DCE Security
: Forthcoming X/Open Preliminary Specification, X/Open DCE: Authentication and Security Services (ISBN: 1-85912-013-X, P315).

Procurement Guide
: X/Open Guide, September 1992, Defining and Buying Secure Open Systems (ISBN: 1-872630-61-8, G206).

XSH, Issue 4, Version 2
: X/Open CAE Specification, August 1994, System Interfaces and Headers, Issue 4, Version 2 (ISBN: 1-85912-037-7, C435).

Chapter 1

Introduction

This chapter explains the importance of security in information systems. It also explains the purpose and scope of this guide in the development and use of security technologies.

1.1 The Importance of Security

The security of information systems used in business is one of the highest priority requirements of users. Since 1991 the Xtra Process (X/Open Company's ongoing market research programme) has consistently shown security among the highest priority issues raised.

As businesses become more dependent on computers (and more distributed) there is a growing appreciation of the need for security. Increased publicity highlighting successful and damaging attacks from viruses and hackers has enhanced awareness. Such attacks often make use of the entry points offered by distributed businesses (and therefore systems). Network connections or dial-in access to a system are typical entry points that can cause problems.

The **Procurement Guide** (see **Referenced Documents** on page xvii) provides essential advice on which types of secure open systems are appropriate to specific circumstances.

1.1.1 Security of Information

The most important asset that most businesses hold today is information. This could be information on customers, contracts, personnel, products, procedures or marketing. The security of this information is normally entrusted to business managers, many of whom have little experience in the control and protection of information. These managers may have other responsibilities which divert them from security issues. Another factor is the speed of change of both technology and the security threats that face business Information Technology (IT) systems. This creates a difficult environment for the average business manager, particularly when using distributed systems (such as workgroup office automation, database management and email), where several sites exist in place of what was previously a centralised data processing facility.

Distributed systems provide useful features. For example, a failure at a centralised computing system is usually catastrophic, whereas in a distributed computing system, a failure at one site need not be disastrous, because work can be performed at the other sites. However, the distributed system also provides an increased security hazard. Each site could potentially have a security risk similar to that of a centralised system, as well as the problems introduced by connecting to a network.

Products are available that can withstand such threats, but the majority of them have been designed to meet specific criteria originally intended to address U.S. Department of Defense requirements. A common complaint in the commercial sector is that these criteria, and hence the products designed to meet them, do not adequately address the needs of business. The products available are not matched to corporate security policies. An example of this is the military desire to prevent disclosure at all costs, as opposed to protecting data from change by an unauthorised person.

Many commercial users do not fully understand their security requirements, or the technologies available to counter threats. Frequently they procure systems with security features suitable for military uses. One of two things may happen:

- The system is delivered with none of the security features turned on, or is configured (from a security point of view) in an unsuitable way. Although it is capable of withstanding the threats to the business, a naive user operates the system as though it has no security features whatsoever.
- The user turns on all the available security features in the belief that this is best. He observes an unacceptable degradation of performance and is under pressure to remove the security protection.

1.1.2 Security Protection

Security is intended to protect an information system from unauthorised attempts to access information or to interfere with its operation. It is concerned with ensuring that the system can meet the following security policy goals:

Confidentiality
Information is disclosed only to authorised users.

Integrity
Information is modified only by authorised users, and only in authorised ways.

Availability
Use of the system cannot be maliciously denied to authorised users.

Accountability
Users are accountable for their security related actions.

To have these qualities demands that security be a consideration in all components of the system.

1.2 Secure Open Systems

To many people, the phrase *secure open systems* seems to contradict itself. Indeed, there is a general misconception that open systems are less secure than traditional vendor-specific systems. This view is encouraged by the default configuration of most systems as delivered, or by the relaxed manner in which most systems are administered and operated. There have been several cases of security breaches of open systems installations, for example:

- The 1988 Internet worm was a malicious program that replicated itself throughout the Internet network.
- Early in 1994 password sniffing attacks by unauthorised users resulted in passwords being obtained.

This does not give a true impression of open systems today. In many cases, the vulnerability of open systems is caused by the failure to apply protection mechanisms sensibly. Any computer system, whether open or not, is vulnerable to attack if it is not managed responsibly. Responsible and sensible operation of the IT system, with respect to security, comes from the correct implementation of a well thought out security policy. Open systems are the basis for some of the most secure installations today, for example, UNIX systems with military grade security have been implemented and are in use.

In today's typical business computing environments, the security of open systems is not the main cause for concern. Instead, the personal computer (PC) causes the greatest concern among users, particularly where PCs are connected to a local area network (LAN).

1.3 Standardised Security

A business user might think that having the same security measures as everyone else would make him more vulnerable, because everyone knows about it. Surely the business would be more secure by adopting technology that no-one else has?

This type of thinking is generally called *security through obscurity*. It introduces a number of problems for most businesses, since various assumptions have to be made, for example:

- The user is able to develop all the relevant technology himself, or it can be supplied exclusively to him from the vendors.
- The user never has to interact securely electronically with anyone else.
- The user has the resources to ensure that the technology is available on all the IT systems used in the business, in a timely manner.

In fact, most commercial enterprises are interested in doing business, rather than developing and maintaining secure business software.

Business IT systems are becoming increasingly interconnected to communicate invoices, payments, correspondence, designs, planning information, orders and products. Anything that can be written, drawn, photographed or recorded is being sent from one computer to another. These computers need to communicate with one another and to interoperate.

There is at least one other problem associated with the security through obscurity philosophy: unless you employ security experts to develop your technology, you are unlikely to be able to gauge the strength of the security measures. As a result, you may be operating your business with technology that has gaping security holes in it. A safer way of working would be to base your IT systems on technology that has been developed in an open, public forum. If the technology is still able to resist attacks after the details of its operation have been scrutinised in public, you have increased confidence that even if a potential attacker knows the technology you are using, he is unlikely to be able to breach the security barrier provided.

An example of how this method has worked in practice has been the Data Encryption Standard (DES). The algorithms for this encryption standard are publicly available, and have so far resisted any attempts to break them. DES is therefore accepted in the international banking community as a standard mechanism for the protection of information. Another encryption algorithm whose details are publicly available, and has also resisted all attempts to break it, is RSA (named after its authors: Rivest, Shamir and Adleman).

Just like any other technology, security needs standards for businesses to reap the maximum commercial benefits. The benefits to any commercial enterprise of investing in secure information systems based on open standards are:

Not Dependent on Proprietary Technology
: The business is able to select a system according to the relevant criteria of cost, applicability, service, product availability and so on, rather than being tied to systems that support a particular proprietary technology.

Portability
: Business applications are more easily supported on computer systems that conform to open systems standards. This leads to reductions in software costs and staff training costs.

Interoperability
: Open systems specifications promote interoperability, enabling the business to combine a variety of computers in a flexible manner to support the business operation.

Scalability

The ability to support small or large systems is inherent in products based on open systems standards. The specifications and technologies upon which open systems are based are intended to support a wide range of computer and network architectures.

The use of open standards allows a commercial enterprise to adapt to a changing business environment in a timely manner. IT system architectures can be designed to suit current and expected business requirements, making best use of the wide variety of offerings available from multiple suppliers.

1.4 Security Framework

1.4.1 Objectives

The principal objective of the framework is to provide guidance on system security to those responsible for: the development of X/Open specifications, future X/Open Security Working Group (SWG) work and application development.

Other objectives of the framework are:

- to support X/Open's mission to produce open systems specifications that promote portability and interoperability
- to satisfy the current requirements (identified by the X/Open Security Requirement Topics Group, for security in distributed systems) and accommodate expected future requirements
- to allow the implementation of different, IT system security policies and security architectures — the system security architectures are appropriate for a single processor through to large, heterogeneous distributed systems
- to identify and specify the common system security services that must be deployed, and describe how these services are to be integrated to provide effective system security architectures
- to guide those defining X/Open components and profiles on how to contribute to the development of integrated security solutions, and how to satisfy the security requirements and obligations that are best met in their specific areas
- to allow for the evolutionary nature of IT system security needs, standards and solutions.

The framework makes use of the concepts defined in **Referenced Documents** on page xvii and is aligned with ongoing work in the IEEE POSIX Security Working Group (P1003.22).

1.4.2 Scope

This document defines a framework for distributed systems security and identifies required security service primitives. This document provides general guidance for incorporation of security qualities into other services. However, this document does not prescribe security models or security data formats for specific application areas. Future X/Open specifications will be able to take advantage of these security services for the development of secure distributed open systems.

This framework is not restricted to any type of security device: both system-based and network-based services are, for example, within the scope of this document.

Chapter 2

Overview

This chapter introduces the concepts of the security framework, and provides references to sections of the document where the relevant concept is explained in more detail. By reading this chapter you can appreciate the essential aspects of the security framework and its objectives without having to read the detailed sections completely.

This chapter outlines security responsibilities within an enterprise and describes how this maps to IT systems. It explains how security information is established and propagated. It then discusses network security and associated issues, and the framework strategy.

Throughout this chapter new terms are described briefly in the text, so it can stand on its own as an overview of the framework. Detailed definitions are provided in the glossary.

2.1 Security Responsibilities

An enterprise is responsible for the safe custody and maintenance of the information on which it relies to manage and conduct its business; in fact its business may be the provision of information.

The responsibility for the protection of the assets of an enterprise rests with the board of directors, or equivalent, who are therefore the ultimate security authority for the enterprise. The security policy for an enterprise details the security objectives of the enterprise. These are then expressed in terms of the security-relevant assets of the enterprise, who is responsible for them, and how they should be accessed and managed.

The board of directors may delegate to heads of departments the authority and responsibility for enforcing specific aspects of the enterprise security policy that apply to the assets under the control of the department. When those assets consist of information held on IT systems, aspects of security policy are further delegated to the IT systems themselves. This is illustrated in Figure 2-1 on page 6.

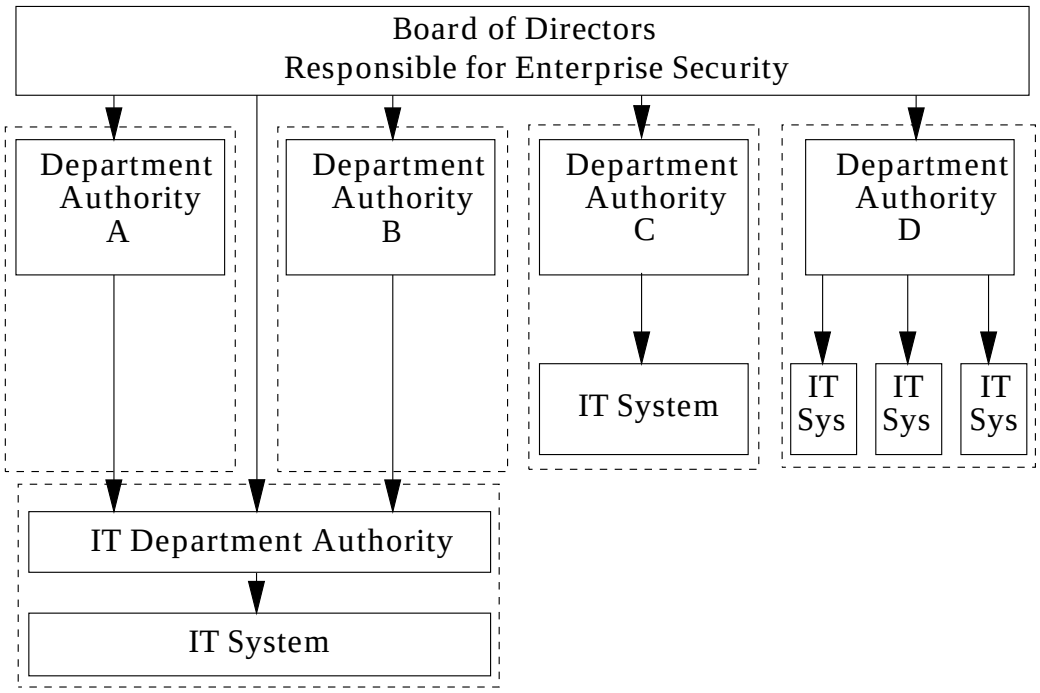

Figure 2-1 Delegation of Authority within an Enterprise

Authority is delegated to four departments. Departments C and D further delegate authority to IT systems under their control. Departments A and B both delegate authority over some of their information and operations to the same IT department, which in turn delegates authority to an IT system. Further information on enterprise security is given in Section 3.1 on page 19.

2.2 IT System Security

Based on the security requirements defined by the enterprise and the local needs of business management, the security policy for IT systems is determined by those responsible for delivering the required level of IT services. The IT operational requirements determine the architecture for the IT systems, including the provision for security. This is shown in Figure 2-2.

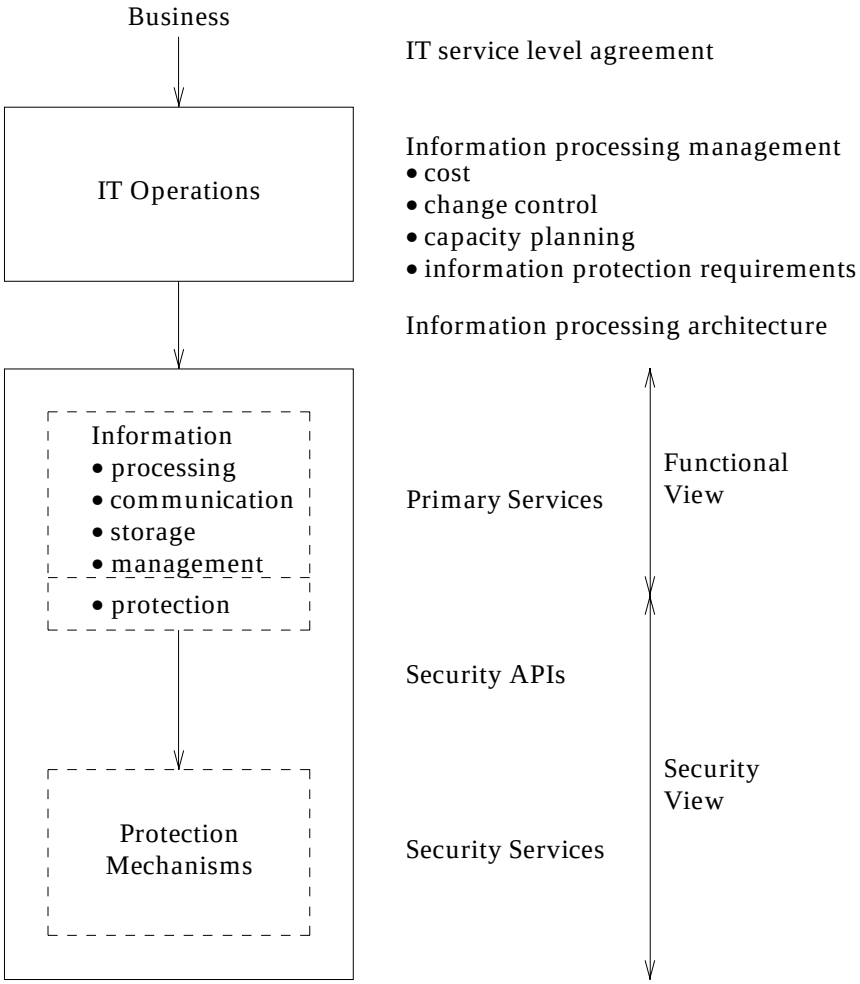

Figure 2-2 Security within IT Systems

Security in an IT system is included in all Primary Services, such as the operating system, communication, data management, and so on. This framework identifies the APIs needed to separate the Primary Services from the security services invoked (and the specific protection mechanisms which underly them). The *security authority* is responsible for defining and implementing a security policy.

A *security domain* is a set of data and related operations subject to a common security policy.

The basic security concepts are described in more detail in Chapter 3.

2.3 Operating Scenarios

In the discussion of security frameworks and services that provide support for secure operation, two common operating scenarios are considered:

- *standalone*
- *distributed.*

A network of systems can support applications which may require facilities such as common remote file server, file transfer, email-type message transfer and remote login capability. In general, these applications require application or platform service security support for application to application interworking.

2.4 Standalone System

The security model for a basic standalone, multi-user computer system is illustrated in Figure 2-3.

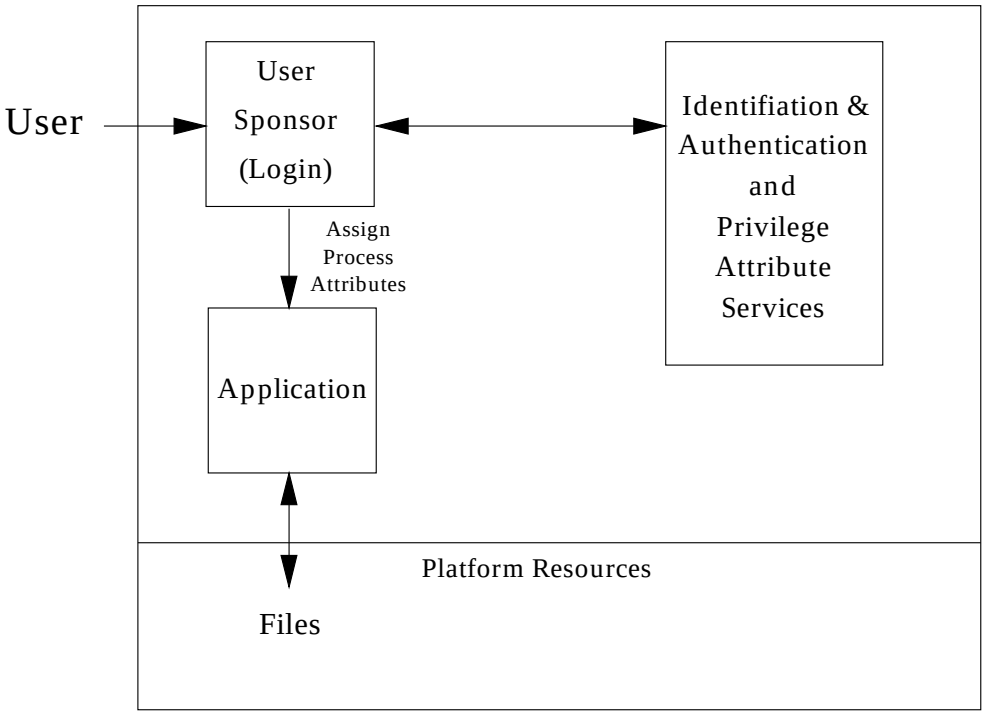

Figure 2-3 Security Model for Standalone System

2.4.1 Connecting a User

To interact with an IT system, a user must establish a relationship with a *sponsor*. This is typically known as a user sign-on (or login) procedure resulting in a *user session*. Often this procedure requires the user to specify a name (*identification*) and provide some proof of identity, such as a password (*credential*). The *authentication* of the name and password must be satisfactorily completed before any further interaction with the system is permitted.

The user sign-on procedure is described in more detail in Section 6.2 on page 105.

Those authorised to use the system are known as *principals*. Further information is given in Section 5.3 on page 69.

A user's access rights within the system (*privileges*) are represented by a set of *security attributes*. Within this framework security attributes defining access rights are referred to as *privilege attributes*. For more information, see Section 6.5 on page 125.

If authentication is successful, appropriate privilege attributes (such as user identity, group identity, capabilities) are assigned to processes run by the user. The establishment of the privilege attributes is separate from the authentication procedure. In some systems an authenticated user can operate with several different sets of privilege attributes, each set being the minimum necessary for a particular role. This is known as the *principle of least privilege*.

Note: Other security information may also need to be associated with the principal, for example, an audit identity. All security-related information associated with the principal is referred to as privilege attributes.

2.4.2 Security Domains

A standalone system can be looked on as a set of interacting security domains. The operating system and the operations and data it supports (for example, access by processes to files) are governed by the security policy applicable to the operating system domain. Within this framework an operating system domain is termed a *platform domain* (see Figure 2-4).

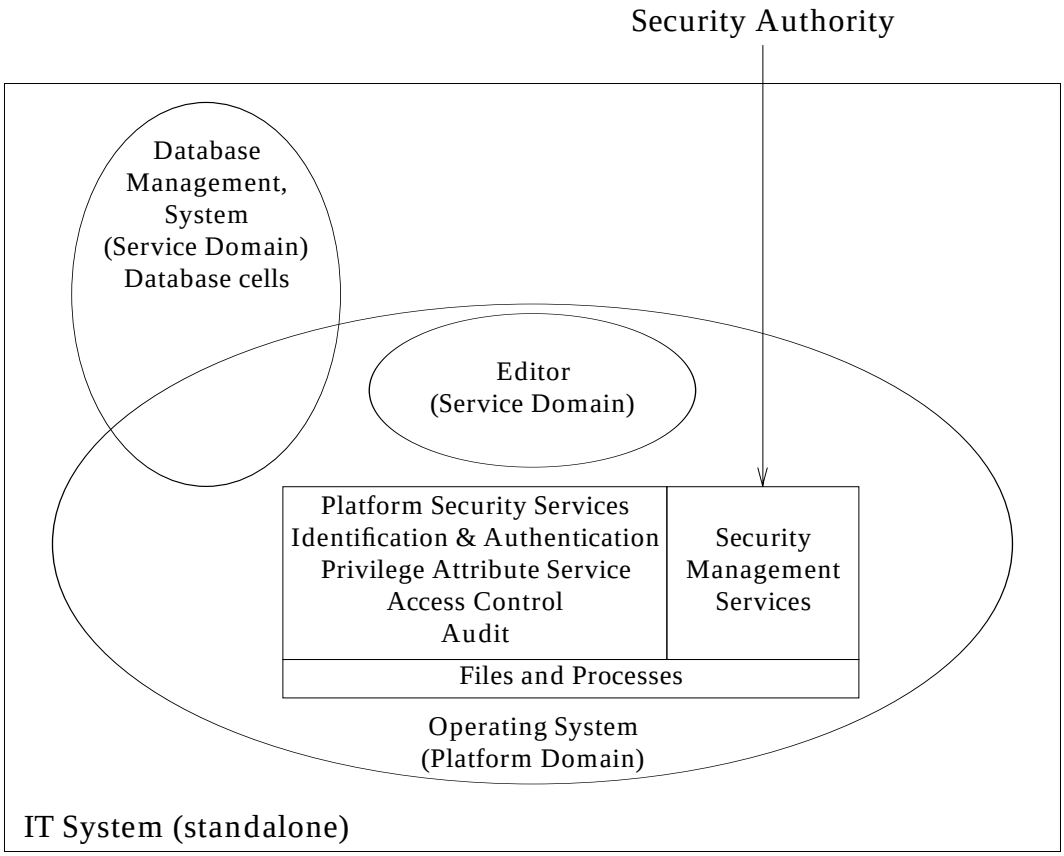

Figure 2-4 Domains in System

Applications are represented by threads of execution (for example, processes) of the operating system and use its resources (for example, files). An application can also be considered a security domain supporting its own set of operations and data at a finer level of granularity than the operating system, for example, a database management system. These are subject to an application-specific security policy. Within this framework an application security domain is termed a *service domain*. In Figure 2-4 interaction between the platform domain and service domains is indicated by the overlapping areas.

The security authority is represented within the system by the security services and the security information which determines the policy enforced. The security authority of the organisation using the system configures and controls the security behaviour of the system using the *security*

management services.

2.4.3 Authorisation

A multi-user system typically controls access to system resources. Normally, the operating system provides the authorisation service and its enforcement. Other service domains may use more complex authorisation processes.

A *protected subsystem* is a service domain enforcing an application-specific security policy. Users can only access the data by means of the programs that implement the operations and enforce the security policy of the service domain.

2.4.4 Propagation of User Session Privilege Attributes

The privilege attributes established for each user must be associated with all the user's operations within the system. Also, the integrity of those attributes must be maintained. For example, a user's operations within a system result in a *tree of execution* within the user session (a process group or session group). Typically this is achieved by *inheritance* of *process attributes*. For example, in UNIX systems, the original user-id and group-id are inherited throughout the tree (unless the set-user-id and the set-group-id mode bits of the executed process image file are set).

A service domain, such as a database, may provide services to many users and control access to its internal data in a manner similar to the operating system. To achieve this the service domain must determine the privilege attributes of the user invoking its services. This can be done by security services implemented within the service domain itself, for example, by making the user sign on specifically to the service domain. Alternatively the privilege attributes already established by the original login procedure can be retrieved.

2.5 Networked Systems

When systems are connected to a network, users of one system can access the services of another. Connection to a network introduces additional security threats through the communication media and increases the potential number of access points to the system. This is illustrated in Figure 2-5.

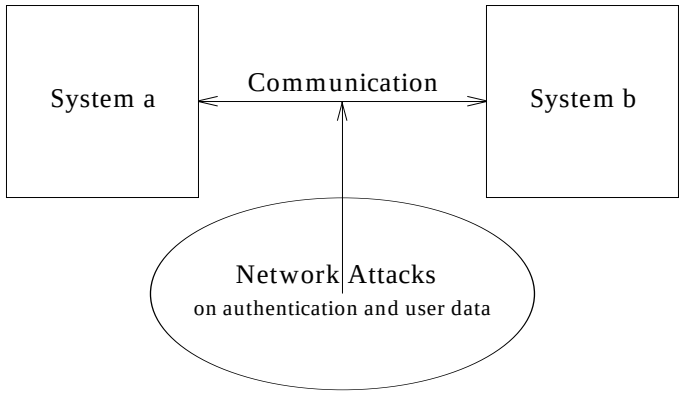

Figure 2-5 Network Security Threats

Network attacks on authentication include *masquerade* and *replay*; network attacks on user data include data modification and *eavesdropping* (see Appendix A). Applications storing or transferring data within a single platform generally rely upon the platform to protect the integrity and confidentiality of that data. This reliance is based upon the physical protection of the platform together with the separation mechanisms of the hardware and software comprising the platform. However, if that data is transferred between platforms using communication media, the same level of protection may not be available. Then cryptographic services may be needed to protect the data. Note that network-based security devices, such as firewall routers, can be used to increase the level of protection offered by the communication media.

The type and strength of the cryptographic services used for this protection are termed a *quality of protection* (QOP).[1] Communication security services can be provided by the platforms in the lower layers of the communication services (the transport provider). Alternatively, these services can be provided within the applications in the upper layers of the communication services.

Applications might need to specify the QOP. The communication service layer within an application must be able to determine whether the platform communication services support the required QOP. If not, the application communication service layer must provide the protection needed.

1. If the communication media are adequately physically protected, such additional measures may be unnecessary.

2.5.1 Types of Networked System Security

When systems originally designed to be standalone are networked together, the security of the network is *platform-focused*. This means that each system is independently responsible for identification, authentication and controlling access to its resources, effectively continuing to act as a standalone system.

Distributed system security is designed from the outset to support a network of computer systems. Each system delegates the control of access to its resources to a common set of security services.

2.5.2 Platform-focused Security

Figure 2-6 illustrates security for a networked system.

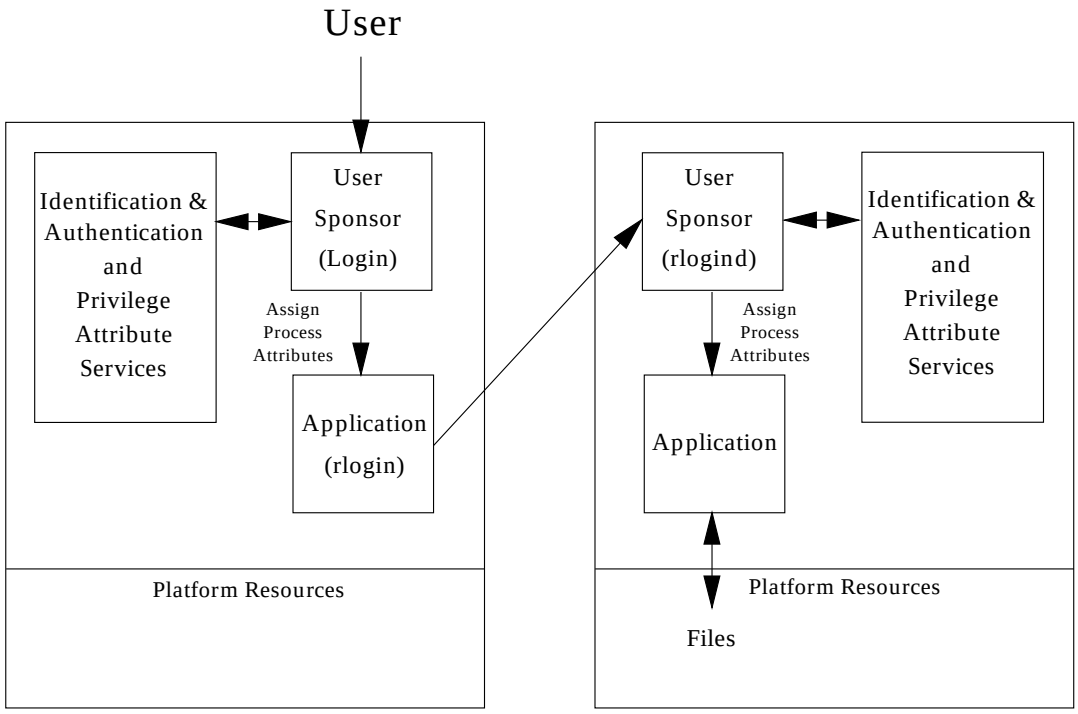

Figure 2-6 Networked System — Platform-focused Security

The essential feature of platform-focused security is that a point to point trust relationship must be established. Therefore a user logged into one system must go through another login procedure to use a second system. Hence management information required to verify authentication information (such as passwords) must be stored on each platform in the network.

The application and the administrator on the target system have to place trust in the application, system software, the system administrator of the initiating system, and in the intervening network to maintain the integrity and confidentiality of the security information being exchanged.

Identities and privilege attributes on one system can be mapped to identities and privilege attributes on another.

Mapping requires the storage of security information (for example, passwords) on the systems.

Note: This is a practice that could increase vulnerability due to the need to present user authentication information (such as passwords) at the entry point to each system. Such vulnerabilities can be mitigated through the use of one-time passwords.

The management of authentication information for each possible user presents an administrative overhead that limits the scalability of platform-focused security to large numbers of networked systems.

2.5.3 Distributed Systems Security

The key feature of distributed systems security is that each system shares trust in a set of common security services. Authority for managing the authentication and privilege attributes assignment for users is delegated to these security services. This is illustrated in Figure 2-7.

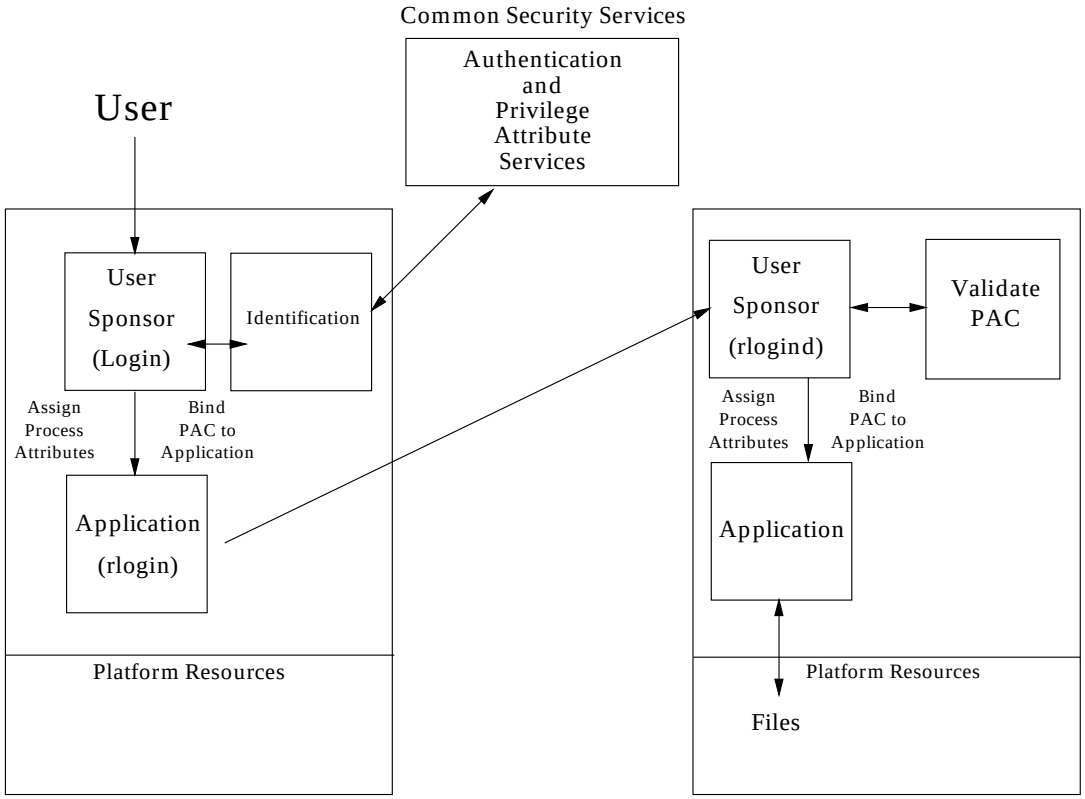

Figure 2-7 Networked System — Distributed

Use of common security services such as authentication and privilege attribute services entails generating and propagating security information, which must be protected on an open network. There are a number of mechanisms available for achieving this, all of which use *cryptography* (the data is transformed to hide its content, protect its integrity or authenticate its origin). One mechanism is that the security information is handled as *Privilege Attribute Certificates* (PACs). A PAC may be cryptographically protected.

2.6 Issues

The primary issues arising from connecting computers are the need to:

- protect information exchanged between the systems
- share and exchange security information between the systems
- trust the security capabilities of other systems and their administrators.

There are different approaches to these needs, generally offering a tradeoff between cost and effectiveness. The actual measures used depend upon the perceived threat and the technology available. Thus systems are usually connected without distributed security services for example, to the Internet, because there is no perceived threat, or the technological support is not available, or the cost of addressing the threat in both financial and usability terms is too high.

The two main benefits in the use of distributed systems security are:

- the protection against network attacks
- the reduction in the numbers of user account passwords to manage.

One potential drawback is that more systems (and people) must be trusted.

2.7 Rationale for Developing a Security Framework

There is a wide range of threats and countermeasures to be considered when addressing the needs of IT system security. At present, effective solutions employing many of the measures mentioned in this document are either not available to commercial off the shelf technology or, if available, they are not easily integrated to provide distributed system-wide solutions. It is difficult for vendors or purchasers to justify the development of customised security solutions.

Many applications try to meet distributed system security requirements within their own domain of control. Consequently, different (and often inadequate) solutions to the same security requirements are devised and offered. There is a need for a structured, consistent set of system security services that are:

- deployed at the appropriate points in the distributed system architecture
- implemented by a minimum number of trusted components
- relied on by the higher-level application services to meet their security requirements
- easily managed
- scaleable to meet different distributed system sizes and configurations.

2.8 Security Framework Strategy

The X/Open security strategy is to provide the necessary foundations to enable migration from platform-focused to distributed open systems. The approach to doing this is to define generic security APIs that are independent of specific underlying security mechanisms. Figure 2-8 illustrates an application programmer's view of the X/Open Distributed Security Framework as a block diagram.

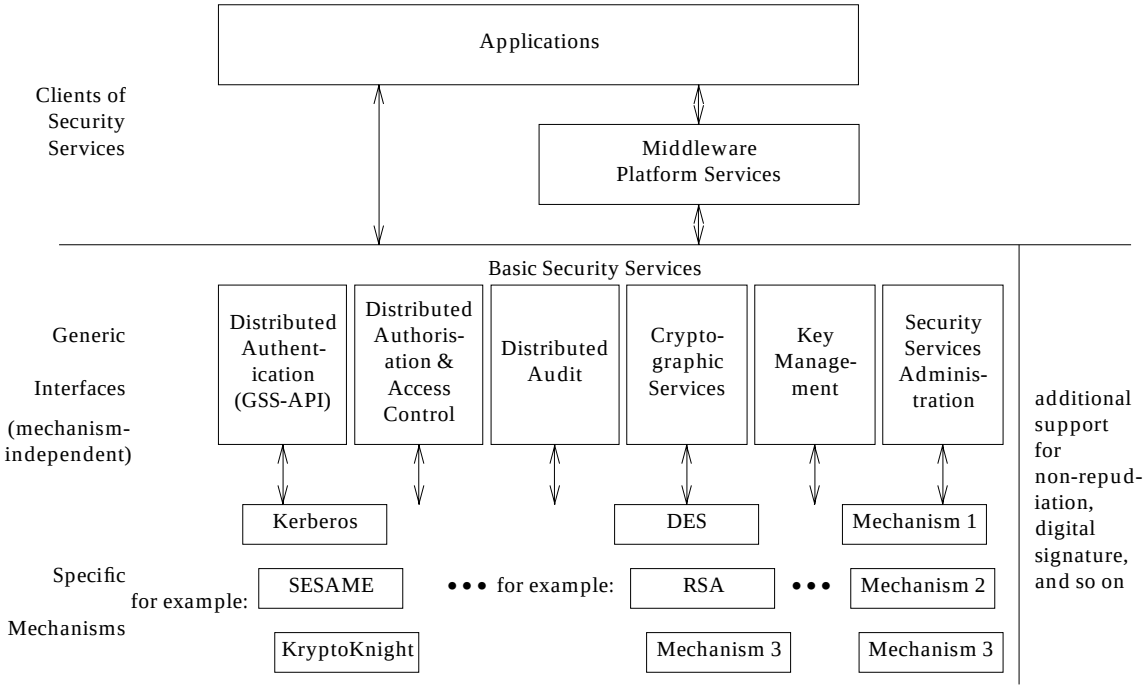

Figure 2-8 Framework Strategy

The block diagram presents three sections:

- The first section consists of the callers of the security services. These are commonly applications and other services required by the application to carry out its functions, such as communication, database or transaction processing services for example. To ensure secure operation, both of these types of caller make use of the security services provided by the distributed system.
- The second section consists of the security services themselves. These are provided through the provision of generic interfaces. In other words, the interfaces are not specific to any particular method of providing the service, but are able to serve as the interface to any method or mechanism. The primary objective of having a generic interface section is to insulate users of the services from specific technologies that may change.
- The third section consists of specific technologies or mechanisms which are accessed by means of the generic interfaces in the second section. Kerberos, SESAME and KryptoKnight are examples of specific technology that can be used by the generic authentication service to support distributed authentication on behalf of the users in the top section. DES and RSA could potentially be examples of specific technology that support the generic cryptographic service. Similarly, all other security services may make use of specific technologies to provide that service in any given instantiation of the framework.

Note: Note that Figure 2-8 on page 16 does not represent any concept of the layering of security services, which may or not be apparent to an application programmer. The concept of layering is discussed in Section 4.7 on page 50.

Each security service must have both operational interfaces for the invocation of security-related operations and administrative interfaces for the control and configuration of those services.

This document also explains how the applications or platform services (clients) can make use of the security services (see Chapter 7).

2.9 Incorporating Security into APIs

Applications can be grouped into those that are aware of security and those that are not.

Security Unaware Applications

Security Unaware clients are indirect consumers of the security services in that they are completely unaware of and unconcerned with security issues. For example, an application that makes use of a platform networking service may use that service without reference to the level of security supported by the platform service. Consequently, the following points should be noted:

- The platform service interface does not contain any security-related information. If the platform provides a secure service, it does so without making it visible to the application.
- The application cannot change, control or direct the provision of security by the platform service.
- The designer of the platform service interface must understand the security needs of applications that use the interface.
- The implementors of platform service specifications must maintain all security functionality without recourse to decision making by the application.

Security Aware Applications

Security Aware clients are direct consumers of the security services; they are required to have control over some or all aspects of the security services. This implies one of the following:

- The client could control the level of security support, for example, through the use of suitable parameters.
- The client could take full responsibility for the provision of security, by handling security-related operations itself, rather than relying on the underlying platform services to provide that support.

2.10 Approach

Any framework of distributed IT system security must accommodate the evolutionary nature of the needs and solutions. It must cover both immediate and longer term requirements.

The development of X/Open specifications for distributed system security solutions is expected to occur in stages, each stage reflecting the evolving requirements and the availability of:

- cost-effective security solutions
- security capable products
- suitable security standards.

This framework uses abstract models of IT system security and general descriptions of threats, vulnerabilities and countermeasures. The framework can be used to identify requirements and how they should be addressed.

2.11 System Security Architectures

This framework provides a generic identification of security services, their component capability and their interactions. It does not attempt to prescribe detailed solutions; these are contained within a system security architecture, which is an instantiation of the framework. The developers of APIs are responsible for defining a security architecture for their system. Some examples are given in Appendix E.

Within a security architecture the general concepts of the framework are all applicable, whatever the nature of the system. However, their relative importance and actual implementation vary with each system.

Chapter 3

Security Concepts

This chapter introduces the top-level IT system security concepts, terms and models used within the framework. It discusses security policy, security authority, security domains and generic threats.

3.1 Enterprises and Security

An enterprise's IT systems can be managed as one or more security domains which enforce specific provisions of the enterprise's security policy.

For an IT system to take responsibility for enforcing aspects of the enterprise security policy, the relevant aspects of the information, policy rules and procedures of the enterprise security policy must be translated or mapped to IT system elements, rules and operations.

The IT system security domains are subdomains of the overall enterprise domain. The enterprise domain is a superdomain of the subdomains. This is illustrated in Figure 3-1. This is also a simplified example of mapping security policy to components in an IT system.

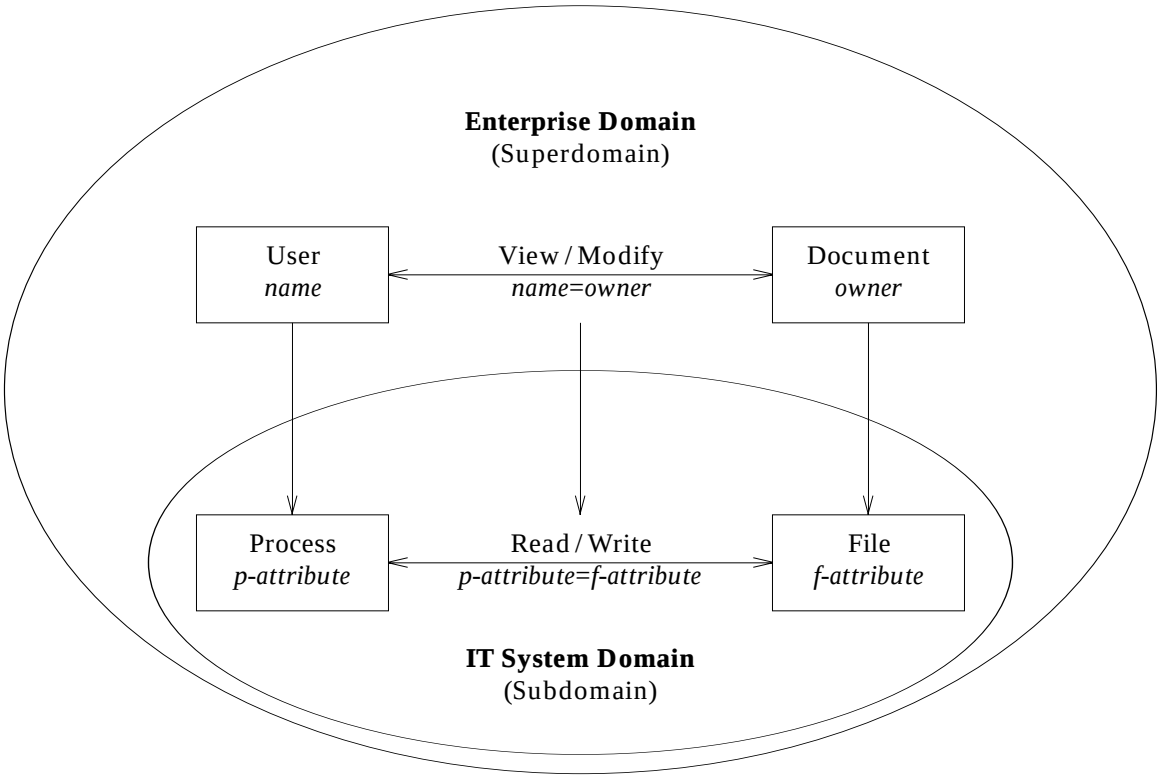

Figure 3-1 IT System as a Subdomain of the Enterprise Domain

Within the enterprise domain illustrated above, the policy states that only the owner of a document is permitted to view it or modify its contents. Each individual has an attribute: a name. Each document has an attribute: its owner. Within the IT system domain, processes

represent users; files represent documents. Each system element is assigned an attribute corresponding to that of the principal it represents in the enterprise domain. Thus, the process attribute identifies the user; the file attribute identifies the owner. The policy rule regulates read and write access by the process to the file, and permits access only if the process attribute equals the file attribute.

The operations of an enterprise involve security domain interactions, both between its own subdomains and between its own security domain and the security domains of other enterprises. These interactions are generally conducted according to agreed procedures and policies. Similarly, interactions between an enterprise's IT systems require supporting security services within the framework of an IT security policy.

The security concepts introduced above are more formally described in the following sections, based upon the definitions in the ISO Security Framework (see ISO / IEC 10181-1).

3.2 Security Policy

A security policy is a set of rules that controls one or more sets of security-relevant operations and one or more sets of entities. A security policy need not necessarily apply to all operations or all entities in a distributed system. So a well-defined security policy clearly states which IT systems, users, operations and data are within its scope.

Within an IT system the operations are the processing operations; the entities are the operation requestors plus the data that is stored and processed.

3.3 Security Authority

A security authority:

- is responsible for the definition of a security policy, and its implementation (where required, as in some cases, the security policy may call for no special action)
- may be a composite entity (such as a committee or a group of system administrative staff)
- may delegate responsibility over some or all of the operations and some or all of the elements within its security domain to one or more entities.

Two security authorities are said to be linked if they are constrained to coordinate their security policies. For example, two security authorities may need to coordinate any changes to the allocation of user identities exchanged between their systems when interworking.

Within an IT system the security authority is represented by the security services supported by the system. In addition a security authority is typically represented by more than one entity within an IT system. For example, a security authority can be an operating system together with a set of security-relevant applications, each of which is responsible for some aspect of the overall security policy or some subset of the elements and operations of the IT system.

3.4 Security Domain

A security domain is a set of entities under a given security policy that are administered by a single security authority for some specific security-relevant operations. The operations of a security domain involve one or more entities from that domain.

A security domain may provide services to principals external to the domain, as shown in Figure 3-2.

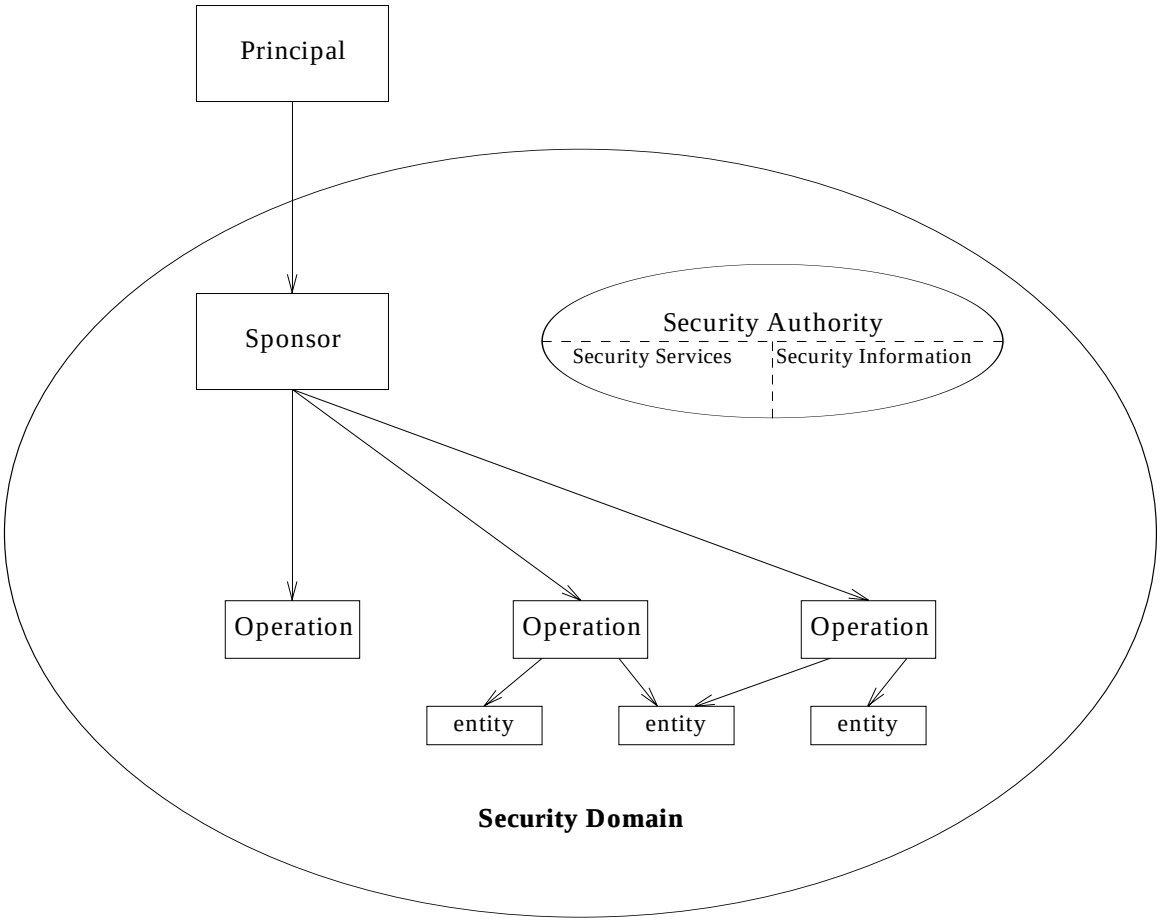

Figure 3-2 Services to External Principals

A domain service is provided by the invocation of all (or a subset of) the domain operations on all (or a subset of) the entities within the domain. Within an IT system a principal might be an individual, or a service within another security domain that acts on behalf of an individual or another service. Within an IT system, an entity within the domain typically might be a set of data (for example, an operating system file), a database table or record, or an email message.

Principals must interact only with a sponsor located within the security domain. The sponsor promotes the principal within the security domain, both facilitating and constraining the principal's interactions with the operations and entities of the domain. The sponsor is responsible for the enforcement of the domain security policy through the direct and indirect invocation of the domain security services. An example of direct invocation is when a user sign-on program sponsors the user to the system requiring the invocation of the user authentication services. An example of indirect invocation is the invocation of an application or

utility program allowing the user to invoke operations on system data. The operations on the system data then invoke the appropriate security services, for example authorisation services.

The security authority is represented within the security domain by a set of security information and a set of security services. The security services utilise the security information maintained by the security domain to enforce the security policy. The security information is associated with the entities and operations of the domain. It is also associated with entities and services external to the security domain, such as principals or external security authorities with which a relationship exists or an interaction has been established.

Management services are necessary to maintain the security information. Security management services and security information are also operations and data of the security domain.

At the highest level, the security objectives defined by a security policy for a security domain are generally to:

- protect the integrity or confidentiality (or both) of its entities and operations
- ensure the availability of, and account for the use of, the entities and operations under its protection.

  Note: The issues of availability are considered on the fringes of the scope of this security framework. To provide a placeholder and also provide further information, availability is discussed in Appendix B.

The security policy of a security domain is responsible for identifying specific threats to the integrity, confidentiality, availability and accountability of the operations and entities of that domain. The security policy must also detail the security measures required to address those threats within that security domain.

3.5 Generic Threats

From the perspective of users and owners, the threats to information and information processing systems are usually grouped as follows:

- unauthorised modification
- unauthorised disclosure
- unauthorised use of resources
- unauthorised denial of service
- false repudiation.

As well as threats arising from unauthorised operations, as listed above, threats also arise from the irresponsible or malicious exercise of authorised operations. It may not be possible to prevent threats arising from authorised operations. However, by recording and analysing system operations it may be possible to deter such behaviour, or react and recover promptly from its occurrence. Examples of threats arising from the exercise of unauthorised operations are:

- authorised modification without audit or change control procedures
- *secondary discretionary disclosure*
- irresponsible authorised use of resources.

Secondary discretionary disclosure is an example of the misuse of access rights; it occurs when a principal authorised to access some information initiates a set of operations that result in access to that information by a second principal who is not legitimately authorised to access that information.

Note: An example is information held in a file to which a set of principals are authorised access by the contents of the Access Control List. One of these principals may copy the information in the file to another file and then set the Access Control List on the new file to authorise access by a different or further set of principals.

Even at this level of definition it is impossible clearly to separate each area of threat. The realisation of a threat in one area often increases the risks of other types of threat occurring. The following sections provide a summary of the individual threats. These are described in more detail in Appendix A.

3.5.1 Unauthorised Modification

This covers the unauthorised modification of information and information processing resources and services.

The concern here is maintaining the integrity of information. The threats are to the procedures and processes involved in ensuring that information is introduced, changed and removed in an authorised manner. Examples are the modification of information relating to: sales and purchase orders, funds transfer, electronic mail and military signals, with the intention of defrauding or confusing.

Realisations of this threat may include the unauthorised modification of the software and hardware base of the information system. This modification may be accidental or intentional; either way it may facilitate the realisation of other threats.

3.5.2 Unauthorised Disclosure

This covers the unauthorised disclosure of information.

Concerns over the confidentiality and privacy of information vary. At one end there are government and defence concerns over the disclosure of national and alliance classified information. For most enterprises, the need is to protect business confidential information. From the point of view of individuals, the primary need is for privacy. This last need is addressed in data privacy legislation that places obligations on the holders of personal data.

Note: Certain classes of system management and configuration information, especially that which is security relevant, constitutes sensitive information that should not be disclosed without authority.

3.5.3 Unauthorised Use of Resources

The unauthorised use of system resources includes the theft of hardware and software components, as well as the unauthorised use of information systems and the services they provide.

Apart from physical theft, the most common concern here is *masquerade*; this means the unauthorised impersonation of an authorised user or other principal. Masquerade involves discovering and using authentication credentials.

The threat of masquerade is not limited to users; malicious systems and services may also attempt to impersonate authorised systems and services.

The unauthorised use of information system resources and services may result in embarrassment and financial loss to the owners, as well as leading to unauthorised disclosure, modification and denial of service.

3.5.4 Unauthorised Denial of Service

Denial of service covers actions and events that prevent information processing systems from providing agreed levels of service to authorised users.

Denial of service threats range from the environmental acts of nature and man, such as fire, flood and bombs, to more directed threats, such as the accidental or intentional swamping of an electronic mail service with a large volume of messages.

The results of different types of denial of service threats can range from catastrophic loss to no more than a temporary suspension of a service. Even the latter could have a significant impact on the business of an enterprise. For example, consider the impact in the stock market of the suspension, for a few trading hours, of some part of an organisation's IT-based trading services.

3.5.5 False Repudiation

This covers the subsequent fraudulent denial by an initiator or a target of having performed some information processing operation.

This category of threat covers many forms of denial of accountability or responsibility. It includes the denial that a message was sent or received. Where such messages are being relied upon, such a denial may damage the interests of one or both parties involved.

3.6 Generic Countermeasures

To meet the security objectives of a security domain and address the threats applicable to the operations of the security domain, a set of countermeasures specific to those objectives, operations and identified threats have to be deployed. The specific countermeasures are a subset of the generic countermeasures described in this section.

Figure 3-2 on page 21 illustrates the generic situation of a security domain providing services to principals external to the security domain. All interactions between the principal and the security domain are by the exchange of information between the principal and the sponsor of the security domain.

3.6.1 Domain Segregation

A fundamental requirement that underpins all countermeasures is that the operations and data of the security domain are segregated from the operations and data of any other security domain and may only be accessed using the security domain's own services by means of well-defined entry points.

Protected Environment

This segregation may be achieved by reliance upon countermeasures provided by the environment of the security domain.

For example, an application may rely upon the physical protection of the hardware and the services of the operating system on which it is hosted, together with the associated system operating procedures to enforce this segregation.

A further example is a physically protected and isolated network, through which parts of the security domain communicate.

Protection Services

In circumstances when part of the environment does not provide adequate segregation, for example, when data is stored or transmitted by means of a medium that is outside the protected environment, specific protection services may be invoked to protect information whilst in storage or in transmission.

These services are referred to as a Data Integrity Service and a Data Confidentiality Service.

Data Integrity Service

The essence of a data integrity service is the inclusion of redundant information within a set of information so that one set of information may be checked against the other set. This is most typically supported by the use of cryptographic mechanisms to generate a checkvalue for a set of data which enables any subsequent modification of the data set to be detected by regenerating the checkvalue and comparing with the original checkvalue included with the information.

Data Confidentiality Service

A data confidentiality service is provided by the use of cryptographic mechanisms to encipher a set of data and protect it from disclosure. Additional mechanisms may be deployed to prevent the inference of information from observing aspects of system behaviour, for example, traffic padding within a network service.

Object Reuse Measures

To avoid the unauthorised or inadvertent disclosure or modification of information arising from the reuse of data storage objects, such objects are required to be purged of information before their reuse to hold data pertaining to another unrelated operation or information set. This countermeasure is required within all services comprising a system. That is, within application services, operating systems and communication services.

3.6.2 Information Verification

All interactions between a principal and the security domain, and also between the elements of the security domain itself, are represented by the exchange of information. The services of the security domain are provided on the basis of the information it receives.

An important class of countermeasure is the verification of the authenticity of that information. These countermeasures require one or more of the following:

- trust in the supporting infrastructure to provide proof of origin and integrity protection
- the existence of an established peer-to-peer trust relationship evidenced by the sharing of a secret, for example a password
- established relationships with, and services provided by, a third party.

Verification of Origin

The verification of information origin may be based upon a connection-oriented identification and authentication protocol, such as user sign-on in the case of a user principal or secure association services in the case of an application principal. This protocol is used to establish and protect a dialogue; that is, a set of messages or session. This type of operation may be referred to as an authentication exchange; it occurs before any operational interaction between the entities.

Alternatively, the verification of origin may be required as a per message service utilising a cryptographic-based data origin authentication service, such as a digital signature, as might be used in a store and forward service such as email. In this case the authentication exchange and operational interaction occur together.

Verification of Integrity

The integrity of information received may be verified by:

Assertion on the Environment
This is trust in the integrity protection provided by the services supporting the exchange of information, for example, the hardware, operating systems and network services, and their physical environment.

Cryptographic Protection
This is the explicit invocation of cryptographic-based checkvalues or digital signature mechanisms by the communicating entities.

Note: This type of protection may be implemented by a lower service layer (for example, network layer) and therefore constitute the *protected environment* asserted by a higher layer component (for example, an application).

3.6.3 Service Mediation

The security domain mediates its provision of service on the basis of the information provided to it and information held by the security domain itself. It verifies the authorisation for the provision of service.

Note: Service mediation is often referred to as access control. Within this framework service mediation is directly supported by authorisation services and access control is considered to be a combination of information verification and service mediation countermeasures. Access control is represented by a combination of authentication and authorisation services.

Service mediation countermeasures may be deployed by several components within a system

Application Measures
Individual applications may verify the authorisation of a principal to invoke particular application operations or to access specific application data.

Operating System Measures
Operating systems typically verify the authorisation for a process to invoke particular operating system functions or to access operating system data such as files.

Network Service Measures
Network services may verify the authorisation for messages to be received or transmitted by network nodes or routed by means of network nodes on the basis of their source and destination, or service type.

Service mediation relies upon security labelling services that bind a set of security attributes with operations and data. This applies to data both in storage and in transmission. The security label may be explicitly bound with the data to which it relates or may be implicitly bound by operational context. For example, it may be represented by security attributes of the storage object containing the data or with the communication association by which it is transmitted.

Security labelling services are inherent within all system components providing service mediation. Thus:

- A database management system (an application) may associate security attributes with fields, records and tables within the database.
- An operating system associates security attributes with processes, files, and so on.
- A communication system associates security attributes with messages and packets.

The information comprising a security label may serve other functions as well as security purposes. Thus, a network packet may be labelled with its source and destination address which are used for routing purposes as well as perhaps authorisation purposes within a network firewall. An important aspect of attributes used as a security label is that their integrity, and the integrity of their binding to the information, must be assured if the security policy is to be enforced.

3.6.4 Operation Recording

This set of countermeasures support the objective of accountability by the detection and proof of security-relevant behaviour and operations. As such these countermeasures represent deterrent measures rather than preventative measures. However, detection of security relevant events permits subsequent (normally management) action to be taken to prevent further damage.

Event Detection

Event detection is a fundamental part of other active countermeasures in that security-relevant events occur within the operation of the other countermeasures, for example, information verification and service mediation failures.

Once security-relevant events have been detected, appropriate action may be taken. This may be one of the following:

- the recording of the occurrence of the event for subsequent analysis
- the issue of an alarm to the security authority or its agent
- the initiation of some security recovery action to prevent future occurrences of the event
- a combination of the above.

Note that combinations of events may constitute a security-relevant event. For example, a security-relevant event could be the occurrence of a number of events of a particular type exceeding a threshold value.

Security Audit

Security audit is the subsequent analysis of recorded security-relevant events, usually off line. This may be used to detect patterns of events that in combination constitute a security-relevant event, or to trace a sequence of events to determine the damage done by any breach of security.

Non-repudiation Service

Non-repudiation services support accountability by preventing a principal from subsequently denying originating or receiving a message, initiating an operation, or disputing the time at which such an event occurred. Non-repudiation may be supported in two ways:

- The recording of repudiation security-relevant events by a third party.
- Notarisation services based on cryptographic techniques to certify the security-relevant details of an operation.

3.6.5 Resilience and Recovery

Availability objectives may be supported by:

- providing for redundancy in the provision of services to protect against the failure of particular components
- enforcing resource allocation rationing to counter the threat of unauthorised denial of service by resource exhaustion
- providing services to permit the timely recovery to a secure state — for example, the disabling of a user account, or a terminal or workstation, for which a configured number of sequential user sign-on attempts have failed.

3.7 Interactions and Relationships between Security Domains

The security-relevant operations of enterprises and IT systems generally require interactions between security domains. These interactions take the form of operations that cross domain boundaries and of elements that are shared by the domains. To establish security management and control of these interactions, it is necessary to formulate a *secure interaction policy*.

A secure interaction policy defines which subset of the elements of each associating security domain comprise the common security domain, and defines the set of permitted operations.

A secure interaction policy is established by the exchange of security information between the security domains. This occurs within two contexts: an administrative context and an operational context.

3.7.1 Administrative Context

In the administrative context, the exchange of security information is persistent. It defines the entities permitted to interact, the permitted operations and the policy that must operate between the associating security domains. An example is the exchange and installation of security information pertaining to the authentication of principals, which may involve cryptographic keys, passwords or address of an authentication service.

If the associating security domains are independent security domains, negotiation may be required between their respective security authorities.

If the associating security domains are both subdomains of a common superdomain, then the secure interaction policy may be defined, and the associated security information established by the security authority of the common superdomain.

Note: It is possible to establish a unilateral policy (for example, anonymous ftp) in which no administrative exchange actually occurs.

In this case, the concept of a secure interaction policy still applies. However, no administrative security information is required to be exchanged but administrative action is required in the target domain. The access control policy in the target domain permits access to the system by any principal, but constrains the operations and elements that can be accessed.

3.7.2 Operational Context

In the operational context, the exchange of security information is transient and occurs during the formation of an association to service a particular set of transactions between the associating security domains. This information may assume a trusted network or may include cryptographically protected certificates, tickets or other data. The term *operational security information* is used in this framework to refer to such exchanged security information in a technology-independent way. The interaction occurs within the context of a secure interaction policy and the security information exchanged is, or is derived from, that established previously by the administrative action that creates and maintains the common security domain, for example:

- A login by a user to an interactive platform session using address-based authentication is only possible if this policy has been previously set up in advance between the machines.
- The generation and processing of a privacy-enhanced email is only possible if both parties know, or can find out, each other's public keys, and each can trust the other's certification.

Note: The lifetime of a security association is not necessarily related to the lifetime of a specific communication service association.

3.8 Assertions on Domain Environment

A security domain's environment includes its parent or superdomain, subdomains to which it delegates authority, and other domains with which it interacts. A security domain does not necessarily provide all the security measures detailed in Section 3.6 on page 25 but may rely upon other security domains that comprise its environment and which provide some of the required protective measures.

If a security policy is to be enforceable, it must specify the the security-relevant behaviour of the domain's environment as a set of assertions. Hence security domains that comprise the environment must satisfy these assertions.

This includes assertions on the responsibilities of principals authorised to use the security domain services, such as not disclosing passwords, and especially upon those principals with administrative responsibilities over the security domain.

Chapter 4

Structuring and Placement of Security Services

This chapter applies the security concepts introduced in Chapter 3 to the development of a generic approach to the definition of a security architecture.

This chapter develops the concept of an IT system as a set of security domains, and shows how the concepts of trust and assurance can provide confidence in the construction of secure IT systems.

Then this chapter describes the structuring of security services within an IT system, both in terms of the classification of security services and their interdependency.

Concepts relating to security in APIs are then discussed, both for invocation and management of security services, and for including security as a quality of IT system services.

Finally, this chapter describes the concept of the layering of security services and of trust boundaries.

4.1 Generic Security Architecture

An information system security architecture is a top level specification that is included within the overall architectural specifications for a system. It covers the high level identification, specification and deployment of a set of interdependent and co-operating security relevant elements (IT services and components of IT services). These elements are collectively responsible for enforcing the system security policy in the system domain.

The security functional elements are variously and collectively responsible for providing: identification and authentication, authorisation, security audit and security administration services.

One of the principal objectives of this framework is to establish a generic system security architecture that supports a range of instantiations. Security architectures vary with respect to:

System Security Policies

The types of system security policies reflect the different types of security functional and assurance requirements demanded by different market sectors.

System Type and Scale

System architectures range from standalone systems to large scale distributed systems required to operate within enterprises and, increasingly, across enterprises. The distributed systems are likely to be heterogeneous, using various system communication paradigms such as: standalone (isolated), client-server (interactive) or messaging (store and forward).

4.2 Security Domains within an IT System

To help bring the various architectural aspects of IT system security down to manageable proportions this framework employs the concept of security domains, each deploying a set of countermeasures specific to the identified threats to its operations and data. Such countermeasures are generally a composition of discrete security services. The concept of nested and interacting security domains and their accompanying security policies can be used to:

- allocate requirements and responsibilities
- address the scaling of distributed systems, particularly those that cross enterprise boundaries
- define boundaries for security responsibility and jurisdiction within systems as well as between them.

4.2.1 Typical IT System Subdomain

An example of subdomains within an IT system is illustrated in Figure 4-1.

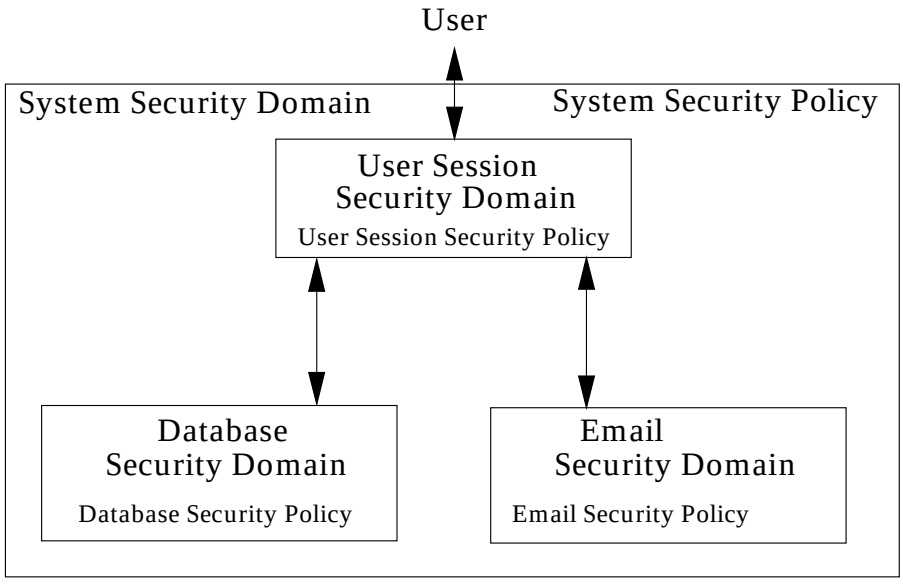

Figure 4-1 Subdomains within a System Domain

The system is subdivided into three subdomains: user session, database and email services. Each subdomain has a security policy applicable to the particular functions it performs. Each subdomain includes security functionality to support its own operations. The database is accessed through a client-server interface. The database client is within the user session security domain; the database application is within the database security domain. The email system is accessed through a messaging interface. The email user agent is within the user session security domain.

The security authority for each subdomain installs administrative security information to support the enforcement of policy within that domain, including the privilege attributes authorised to use the domain's services.

A user interacts with the user session security domain by logging in. A user is permitted to use the services of that domain once the authentication information he supplies has been validated by the user session domain (for example, his identity is authenticated by means of his password) against the administrative information configured by the security authority. See Section 5.3 on page 69 for more information.

A user may need to use the services supported by the database security domain or the email security domain. Each of these security domains requires the creation of an association from the user session security domain to access their services. This is achieved by the exchange of further operational security information. Further user authentication is required, unless the operational security information supplied to the user session security domain can be reused within the database or email security domains. Reuse is possible if there is a distributed security service.

4.2.2 Distributed Security Service Domain

In Figure 4-2 a distributed security service domain has been included within the system domain.

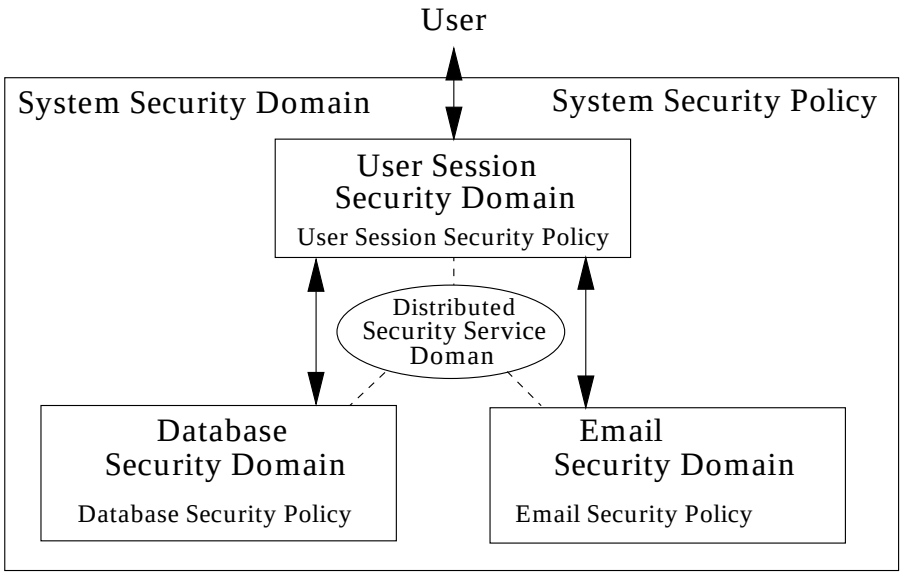

Figure 4-2 Distributed Security Service Domain within an IT System.

The distributed security service domain is a subdomain of the system domain trusted by the other subdomains to provide operational security information. On user authentication, operational security information is issued by the distributed security service to the user session domain, for example a combination of authenticated identity and authorisation information. This operational security information may be used within the user session domain and also exchanged with the database and email domains to establish associations to use their services. The database and email domains now verify the origin of the operational security information instead of re-authenticating the principal it represents. The architectural model discussed here supports a single sign-on environment, such that the user does not have to authenticate to every different system.

4.2.3 Platform and Service Domains

At the highest level, the framework separates the major architectural functional areas into the following types of security domain:

- *Platform domains* provide the services that control and structure the physical, processing and communication resources of an IT system into elements that may be used by the service domains.
- *Service domains* provide information processing services to principals in the context of enterprise operations (applications). The service domains may also comprise a set of *middleware* services that together with the platform services constitute the system infrastructure services.

The infrastructure services collectively provide the system environment that facilitates: communication, distributed processing, interactions with users, and system management and control.

Platform domains structure the physical resources of an IT system, processor memory, data storage and communication media into identifiable elements with assigned security characteristics and which mediate the flow of data between these elements. To enforce data flow mediation successfully the services of a platform domain are presented at a protected, non-bypassable interface.

A platform domain is typically composed of operating system and communication system elements. The actual interfaces and services presented by a platform domain are dependent upon the implementation. For example, they may be represented by an interface equivalent to the **XSH**, **Issue 4**, **Version 2** specification system service interface, such as a traditional operating system kernel implementation, or by a microkernel interface with the **XSH**, **Issue 4**, **Version 2** specification system services implemented as service domains.

Service domains are formed from the elements of platform domains, for example, processes, files and communication channels. Thus inter- and intra-service domain operations involving interactions and data flow between elements of platform domains are subject to the security policy enforced by the platform domains.

The inter-relationship between service domain components and a platform domain is that of subdomain to superdomain. The platform domain enforces a basic policy on the operations of the service domain but can surrender some aspects of policy enforcement to a particular service domain component assigned that responsibility, for example a process that possesses a particular authorisation (capability).

A service domain relies upon the platform domain to enforce policies that provide for its protection, and might rely upon a platform to control its interactions with other service domains.

It is common for a service domain to provide services in support of other service domains, in particular some service domains may provide authentication or security attribute services. The system security architecture must identify such dependencies between service domains.

The point of separation between platform and service domains is based on the current realities of most open system implementations. Typically, a hardware-supported, separation and protection boundary is found at the interface to the operating system kernel. The framework does not prescribe which parts of the infrastructure services lie below that protected interface. Nor does the framework prohibit the use of hardware-supported multi-ring protection domains. In this case the higher layers of the infrastructure and application service domains can be separated and structured using a stronger protection mechanism.

Where the operating system services of several processors are closely coupled and they are all enforcing the same security policy, they can be considered as a single platform domain.

4.3 Distributed Systems and Security Domains

Figure 4-3 shows a distributed system comprising two separately administered IT departments, X and Y. Department X is responsible for Platform Domain 1, which provides the front end for user access to the IT system's services, and Platform Domain 2; Department Y is responsible for Platform Domain 3.

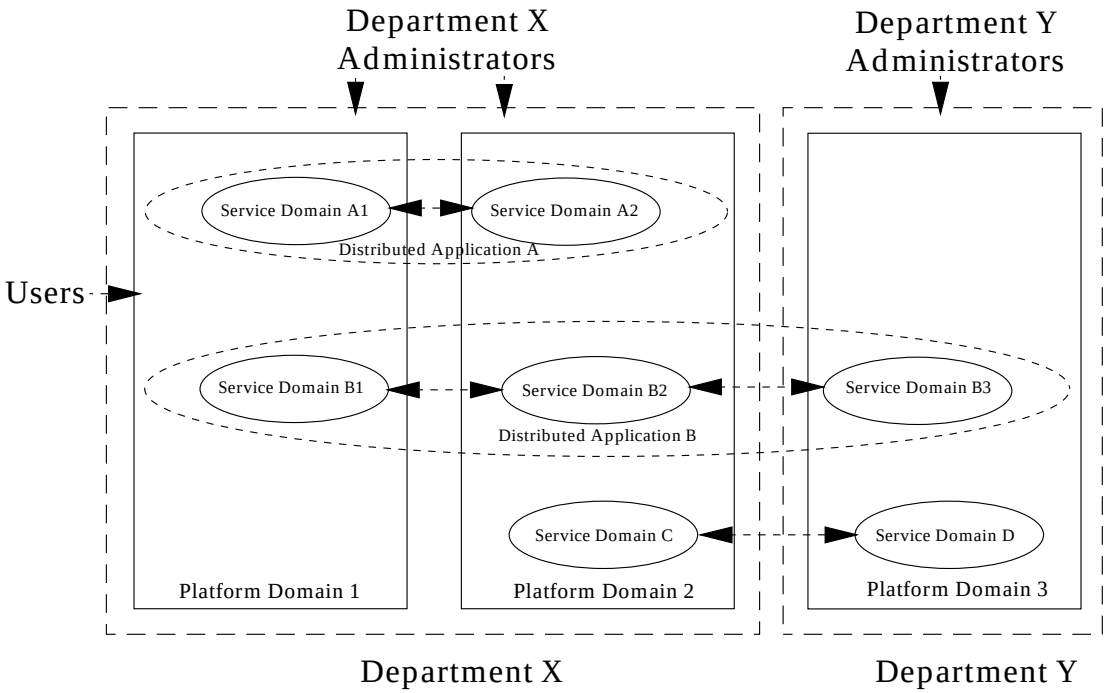

Figure 4-3 Security Domains in Distributed Systems

Although Platform Domains 1 and 2 are physically distinct, they are both part of a single Department X administrative domain.

Each platform within the IT system supports a set of distinct service domains. Some platform service domains are grouped together with service domains on other platforms to form distributed applications, such as the illustrated distributed applications A and B. Examples of such cooperating service domains are components of a directory system (directory user agents and one or more directory service agents).

Each service domain enforces a security policy common to the distributed application.

Other service domains, such as the illustrated service domains C and D, do not form a distributed application with a common policy but represent an interaction between two separate applications with individual security policies. An example might be an application client invoking a database server.

However, the functions of each service domain are subject to the security policy enforced by the platform on the platform domain elements that comprise it. The interactions between the service domain elements of the distributed application are subject to any interconnection policy enforced between the platforms.

A distributed system may therefore be constructed so that some security services are implemented within the platform domains, or within the service domains, or within both.

4.4 Trust and Assurance

4.4.1 Trust

Trust is a fundamental security architecture objective. Trust is defined as follows:

> Element x trusts element y for some classes of operation in the context of a security policy if and only if element x has confidence that element y behaves in a well-defined way that does not violate the security policy, (see ISO / IEC 10181-1).

The administrative actions that establish a secure interaction policy between two security domains form a trust relationship between them. Each trusts the other to behave in a manner that does not violate the secure interaction policy. An example of this form of relationship is a secure interaction policy permitting the exchange of information whose privacy is to be protected. Regardless of any security measures implemented within or over the communication services transferring the information the originator must trust the recipient to uphold the privacy security policy once the information is delivered into the recipient's custody.

Similarly a trust relationship exists between a subdomain and a superdomain and is established by the administrative actions integrating the components of an IT system into a whole. For example this could be the assignment of security attributes, including capabilities, to service domain components, that is, program and data files.

Particular trust dependencies arise between a service domain and the platform domains that underlie it in order to maintain the integrity of the security services of the service domain. As derived from ECMA-138, these are:

- Protect the security services and objects against external corruption. In other words, provide protection for the elements substantiating the service domain against unauthorised access by elements not constituting part of the service domain. An example is the privacy of process address space.
- Provide the communication security services (for example peer entity authentication, confidentiality and integrity) requested when forming associations or indicate that it is unable to do so.
- Provide the necessary initial security information to security services and objects. Examples are cryptographic master keys and identities for service domains.
- Enforce use of an appropriate authorisation service for each access request. That is, the mediation of access to platform data cannot be bypassed.
- Direct all initial human user contact to a subject sponsor and secure this link according to the local security policy. This is may include the provision of a trusted path. Any initial contact by a principal with a security domain must be directed to a component acting as sponsor for the security domain.

4.4.2 Assurance

The security-relevant and enforcing elements are sometimes known as the *trusted* elements. Here *trust* implies that the required degree of confidence (assurance) in the correct and continued working of these elements is established and maintained with respect to the security policy in effect.

Confidence in the overall system security solution is established by designing, implementing and proving a system security architecture against these general objectives and criteria:

- The trusted elements are closely integrated and can thus be relied on collectively to enforce the system security policy.
- The trusted elements are adequately separated and protected from the *untrusted* elements, so that the opportunities for the latter to interfere with the correct working of the former are minimal.
- The trusted elements are deployed such that they can appropriately mediate the security relevant operations of the untrusted elements, and that this mediation is not easily bypassed.

Confidence in the continued correct working of the trusted elements is addressed by the system security architecture design and through the application of well-defined security operating procedures and strong operational configuration management and administration to the system.

Any delegation of responsibility for the enforcement of security to the information system must be based on some measure of confidence in the correct and continuing working of the security relevant measures and functionality.

The procedures for the rating, measurement and achievement of confidence (assurance) are referred to as evaluation. The schemes for specifying and evaluating system security assurance continue to evolve. However, the basic engineering principles and objectives necessary to support assurance requirements need to be addressed in the development of system security solutions and architectures.

Assurance establishes confidence in the correctness and effectiveness of security solutions and their components at each stage in the system life cycle:

- architectural design
- component and product specification
- development
- supply
- deployment and integration
- maintenance and management
- disposal.

4.5 Classification of Services

A service is a combination of functional and data elements that are exercised and accessed through a well defined interface. The range of operations performed by the service functional elements on the service data elements or the passed data elements is specified by the interface.

Security services are classified as illustrated in Figure 4-4. (Note that this figure is not a representation of architectural layering, which is discussed briefly in Section 4.7 on page 50.)

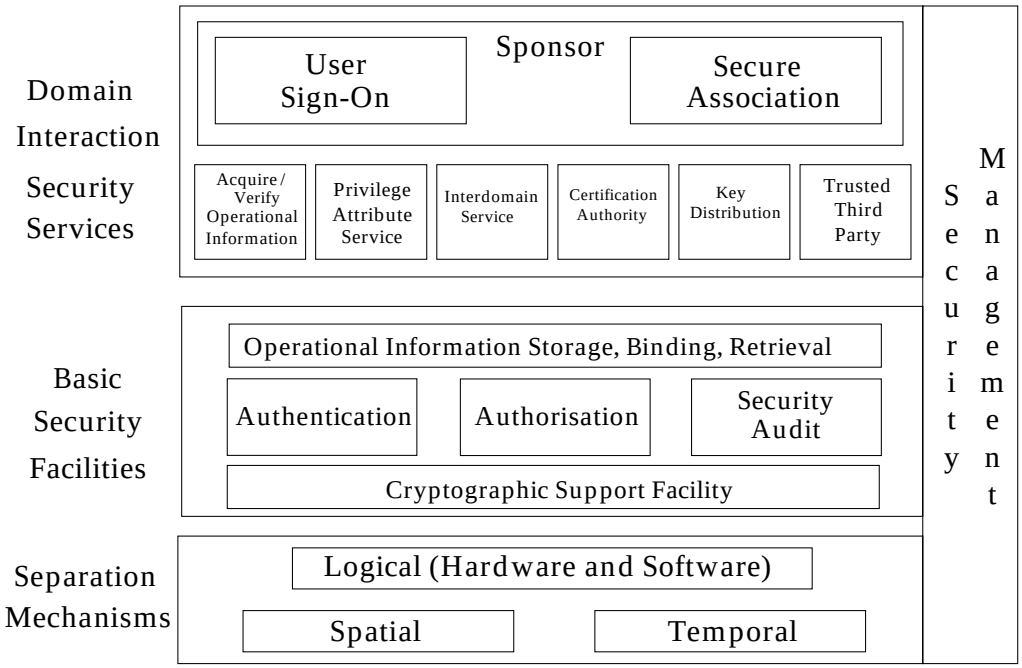

Figure 4-4 Classification of Security Services

The classes are:

- separation mechanisms
- basic security facilities
- domain interaction security services.

Supporting all services are separation mechanisms responsible for the segregation of the system resources used to underlie the IT security domain data and operations.

The basic security facilities enforce primitive elements of security policy.

The domain interaction security services control the security information environment in which the basic security facilities operate. These services are responsible for the distribution and exchange of security information necessary to support interactions between security domains (including users in the physical domain interacting with the IT system).

Requirements for security management are applicable to all classes.

Two additional services are omitted from Figure 4-4. Security services are dependent on them but they are not strictly security services as they have a wider function. These are:

Time Service

Authorisation may be based upon time of day. Security tokens and certificates may include expiry times or lifetimes. For these values to be meaningful and applied consistently within a distributed system requires a distributed time service. This service must ensure that time is synchronised between all the platforms and between all the services within the distributed system.

Name Services

The security services depend upon the accurate identification of principals and data. A principal or data may be referenced directly or indirectly (aliased). For example, a file may be identified by an inode but referenced by a filename within a directory. If security is to be applied consistently, there may need to be enforcement of a policy on name reuse and name re-assignment.

4.5.1 Separation Mechanisms

To protect the integrity and confidentiality of the operations and elements of the domain, a security domain must provide for the segregation of the domain's data and operations from any other domain's data and operations.

In addition, the maintenance of integrity of those data and operations of a security domain that comprise its security services (termed the Trusted Computing Base (TCB) of the domain), depends upon their segregation from other data and operations of the domain.

The physical and processing resources of the IT system underlie the data and operations of its security domains. The segregation of domain data and operations therefore depends upon techniques for the assignment of IT system resources to support domain data and operations, and the protection of the segregation of the resources assigned.

The provision of such segregation may be based upon three protection and separation techniques:

Spatial

As a segregation mechanism, this is the simplest and crudest in that it is based on the physical separation of information processing resources. A trivial example is a standalone single user system. This is a simple and effective technique, but has restricted applicability to the segregation of component processing environments within the context of distributed system objectives.

However, physical protection of processing and communication components is an important factor in determining the nature, strength and placement of security measures within a distributed system.

Temporal

Temporal separation techniques operate at different levels. At the coarsest level of granularity, an information processing system is only used to process information of one sensitivity at any one time and is totally purged before being used again. This technique when applied at a finer level of granularity to targets, such as transient files, buffers and memory address space, is used to counter disclosure through garbage collection and similar attacks. As such it is known as the *object reuse* measure. Another dimension to temporal separation is object locking as used by logical integrity mechanisms.

Note: Single threading and atomic transaction techniques are important to the preservation of the integrity of security-sensitive information.

Logical

Protection and separation techniques of this type are based on the combination of hardware

and software to create separation environments and properties at different layers within an IT system. At the lower levels, the operating system, usually through the use of specific hardware features, separates and protects itself from the vagaries of higher-level capability. At varying levels of strength it provides separation of the different services and resources it offers, for example devices, directories and files. It also separates and protects the higher functional components from each other by establishing a multi-processing environment, controlling the mapping of memory to each process, and mediating the different ways processes can communicate with each other. These operating system protection and separation features are fundamental to the implementation of any reasonably robust system, let alone a *secure* one. Based on their sound implementation at the operating system level, logical separation and protection techniques are used, where required within higher-level functional components.

Deficiencies, or inefficiencies, in the support provided by the above segregation techniques, for example in the use of communication media, may be countered by the use of cryptographic services deployed within higher layers of service, (see Chapter 6). However, note that the security provided by cryptographic services depends upon the above segregation techniques being deployed at critical places within a distributed system to protect the cryptographic algorithms and keys.

4.5.2 Basic Security Facilities

The basic security facilities are security service primitives that provide a discrete security service. These include:

Operational Information Binding and Retrieval
This provides for the binding and retrieval of operational security information associated with principals to a set of operations. Examples are *setuid*( ) and *getuid*( ) in the **XSH**, **Issue 4**, **Version 2** specification, *sec_login_set_default* in the **DCE Security** specification, and *gss_get_attributes*( ) in the **GSS-API Extensions** specification.

Cryptographic Support Facility
This provides various services, cryptographic transformation and key management services, for example, DES encryption and decryption.

Authentication
This provides for the validation of claimed identities, for example, traditional system password mechanisms and *sec_login_setup_identity* or *sec_login_validate_identity* in the **DCE Security** specification.

Authorisation
This provides authorisation for the use of operations and access to data in support of access control policies, for example access of processes to files, or access by a client application to the tables and records of a database.

Security Audit
This provides the detection, recording and analysis of the occurrence of security relevant events, for example, the audit services included in the POSIX.1e specification and in the **Audit and Authentication** snapshot.

The basic security services are described in detail in Chapter 5.

4.5.3 Domain Interaction Security Services

These services support the secure interchange of security information.

The operation of the basic security services depends upon the association of security information with the operations and data of a security domain and with the principals that use a domain's services.

To use the services of a domain, an individual needs to form an association with the domain. The domain interaction security services are responsible for the authentication of a principal's identity and establishment of a set of security information representing the security context within which the domain's services may be provided to that principal.

Such operations may require the invocation of services within other security domains through the formation of associations between the security domains. To propagate the security context within which the association is created and services invoked, security information must be exchanged.

The services to support the formation of such associations between individuals and the IT system, and between security domains within the IT system are described in detail in Chapter 6. They comprise:

Sponsor Services
These services sponsor a principal within a domain.

User Sign-on
This service establishes an association between a user principal and a security domain. Examples are UNIX login, DCE login, ftp, rlogin and explicit services in the **GSS-API Base** specification.

Secure Association Service
This service provides for the creation of associations between service domains (applications) in both interactive (connection-oriented) and store and forward (connectionless) scenarios. Examples are implicit services in DCE RPC, Secure X.400 and TSIX(RE).

Privilege Attribute Service
This service provides sets of privilege attributes. Examples are *getpwent*( ) in the **XSH**, **Issue 4**, **Version 2** specification and the DCE Privilege Server.

Acquisition and Verification Operational Information
This service provides for the acquisition of operational information (such as a security context) and the verification of its authenticity. This is implicit in DCE RPC services and explicit in the **GSS-API Base** specification.

Interdomain Service
This service provides for mapping security attributes between security domains. An example is the token mapping database of TSIX(RE).

Certification Authority
This service issues and manages security certificates. An example is the certificate services related to Privacy Enhanced Mail.

Key Distribution
This service provides for the distribution of cryptographic keys between communicating principals. Examples are Kerberos, SESAME and X.509.

Trusted Third Party Services
These services support non-repudiation and notarisation services.

4.5.4 Security Administration Services

Each security service has both operational and management aspects. The definitions of each security service in Chapter 5 and Chapter 6 include definitions of the management aspects of each service required for configuration and control of the service.

Additional administrative responsibilities may include:

- controlling the starting and stopping of security services
- detecting and handling security service failure and ensuring recovery to a secure state
- facilitating and managing the creation, maintenance and consistency of the distribution of security management information for each security service

  This may include the distribution of cryptovariables (keys) used by the cryptographic services.
- maintaining historic (archived) records of system security configuration

  This information is usually required to support historic audit trail analysis, and may therefore, be included as part of the function of the accountability and audit services.
- collection and management of the current security operational state information from the system

  Here there is an overlap in responsibility between access control and management services. Because this security operational state information is not exclusive to access control services, it has been allocated as a security management responsibility.

4.5.5 Relationship to Platform and Service Domains

The model of the classification of security services does not imply a particular mapping to platform and service domains.

The logical separation mechanisms available are generally enforced by platform domain mechanisms, but service domains are also responsible for applying separation measures to elements of their own domain, for example, object reuse.

The basic security facilities are deployed within both platform and service domains and some domain interaction services may be supported by service domains, platform domains or both.

4.6 Security Service Integration

Systems are composed of many different services that provide facilities for information processing. These services can generally be considered independently of each other, for example, system services and communication services. These services are referred to as *Primary Services* within this framework. Generic qualities such as usability, manageability and security are inherent within all Primary Services.

The security requirements on any particular primary service within a given system are derived from the system's security policy and are mapped to the system's services by the system's security architecture. This section presents a model for operational and administrative security APIs, and the integration of security with Primary Services.

4.6.1 Security Service Model

Figure 4-5 illustrates a security service model. The model supports the description of the characteristics of the different security services, their placement within a system and the potential impact on primary service APIs. This model of interaction with security services and their APIs is applicable to services at all levels within a system.

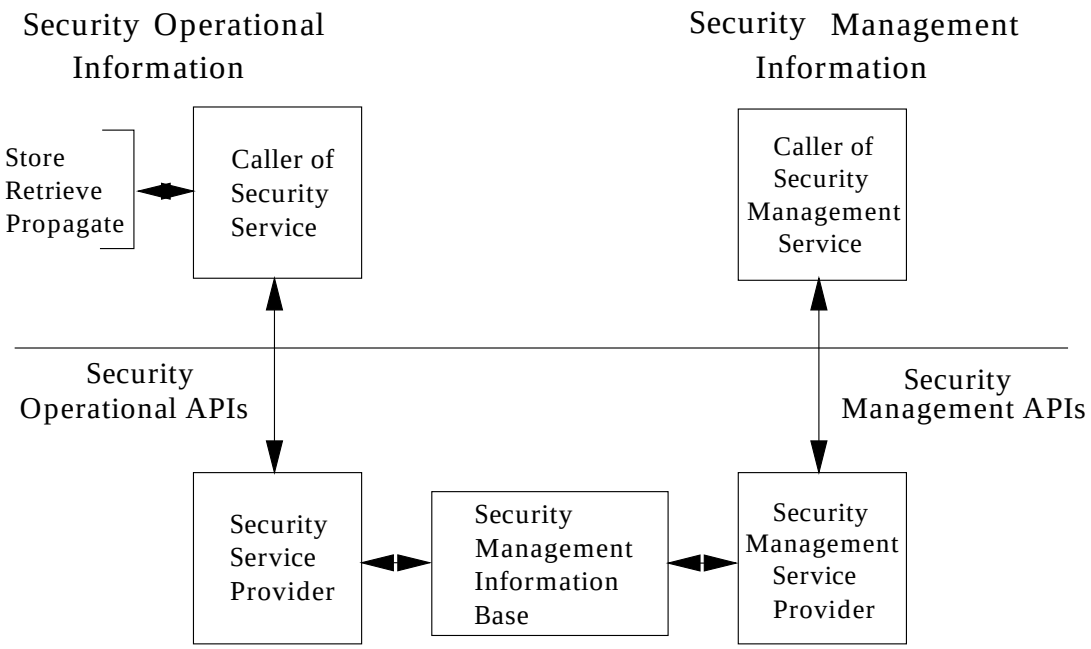

Figure 4-5 Security Service Model

A security service is based on the provision and use of security information. The security information is of two types:

- security management information established by administrative action
- security operational information established by operational actions.

Each security service provides two sets of APIs: security management APIs which handle the security management information, and security operational APIs by which the actual security service is provided.

Security Management APIs

The security management information is manipulated by a set of security management services and accessed by security management APIs. The security management information can be considered to be held in a Security Management Information Base (SMIB), which is accessed by the security service provider as required. The SMIB requires protection from access by any functions other than the security service provider and the security management functions. The integrity and confidentiality of the SMIB must be maintained.

The SMIB is a set of security information associated with the elements of the security domain. The management services relate to the installation, modification, listing of the security information and the enabling and disabling of a particular security service. The security management APIs are used to control and configure a particular security service.

Each security service has a specific set of security management APIs and a specific set of security information associated with each element within a security domain. Therefore, each element may have several sets of security information associated with it, potentially one set for each of the individual security services.

Security Operational APIs

The security operational information is handled by the primary services that call the operational security services. The security operational information is passed between the invoking functions and the security service provider across the operational interfaces to the security services. The operational related services deliver the actual security service.

Operational security information is generated initially as a result of principal authentication. It is then bound to set of operations executed within the system on the principal's behalf. That is, it must be associated with every process or thread created and executed as part of the session servicing the principal. Also appropriate operational security information must be propagated to any independent service that provides services to that principal. The integrity and, where required, confidentiality of the operational security information must also be maintained, including its binding with the sponsor and principal pair.

The current security state of a security domain at a point in time is the combination of the SMIB and the current set of security operational information maintained by that security domain. The security state of a system is represented by the combination of the security states of every security domain within the system.

4.6.2 Security and Primary Service APIs

Primary services required to enforce security need to call on a security service. The basic model of the interaction between primary and security services is illustrated in Figure 4-6 on page 47. Those who develop APIs to these services must consider the *security context* within which their APIs must operate. The security context consists of information associated with:

- the caller of the API (*initiator*)
- the data accessed by the API (*target*)
- the service provided by the API (*operation*).

The security context also comprises the *security policy* rules to which the operation and data are subject (see Section 3.2 on page 20).

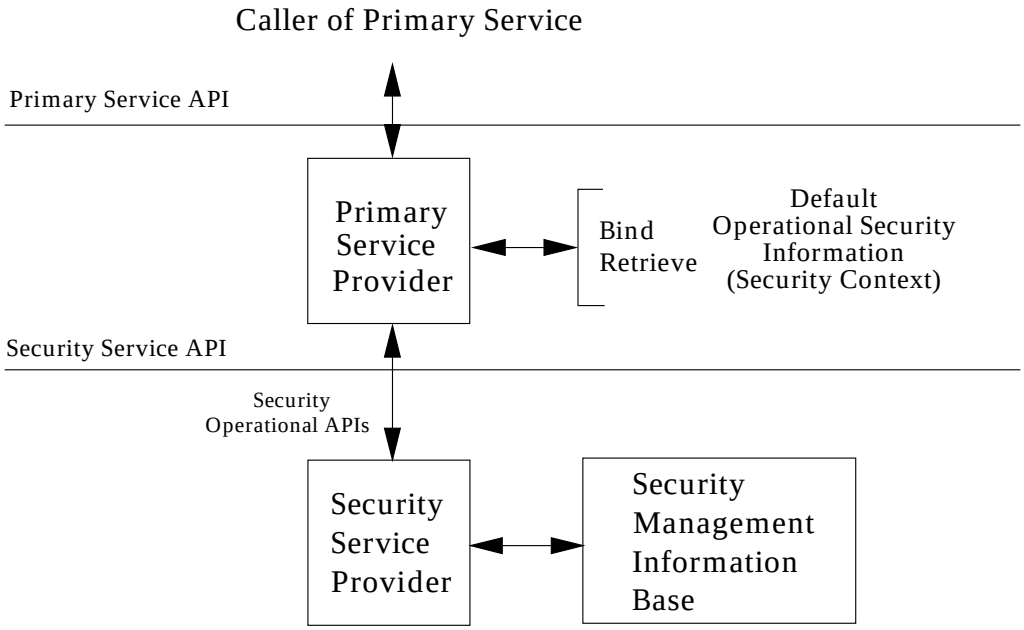

Figure 4-6 Primary and Security Service Interaction

The primary service provider is responsible for invoking the security services using the appropriate operational security information, by means of the security operational API. The appropriate operational security information is generally that within the security context bound to the thread of execution within which the primary service is invoked.

It may be necessary to include operational security information within the primary service API. The requirements of the callers of the primary service can be considered in terms of their security awareness. Security awareness is relative to each individual type of security service. Thus a caller may be aware of one aspect of security, for example authorisation, but not aware of others, for example audit. In addition, the security awareness of a caller for some policies may have a finer granularity than for other policies. Thus a caller may be aware of ACL-based authorisation but not aware of label-based authorisation.

4.6.3 Security Awareness

The security awareness of primary service callers is grouped into:

- security unware
- security selecting
- security enforcing.

This is illustrated in Figure 4-7 on page 48.

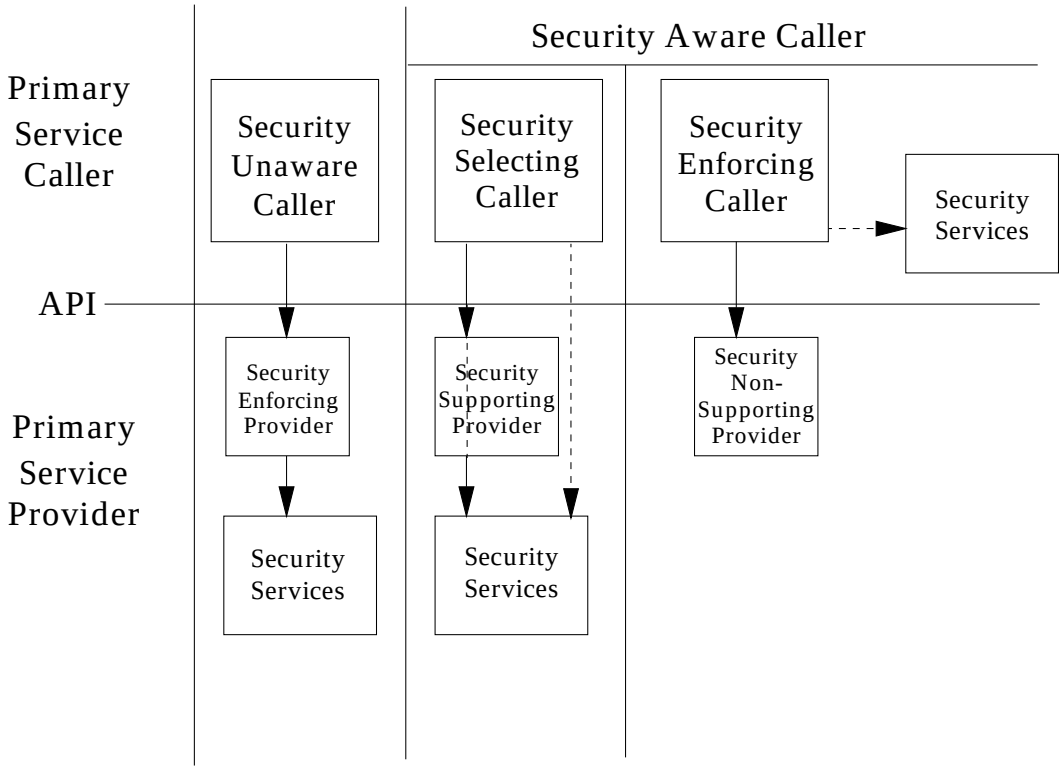

Figure 4-7 Relative Security Awareness and Responsibility

Security Unaware Caller

In this example the callers of the primary service are unaware of security. The primary service is a *security enforcing* provider and includes functions to invoke or implement the appropriate security services, such as authorisation. A security enforcing primary service needs to include functions to acquire and handle any security operational information required by the security service.

The primary service operates within whatever default security context it is invoked. This default security context must have been previously established by security aware functions, for example a User Sign-on service (see Section 6.2 on page 105). In this case the primary service API does not provide security information within its input and output parameters. However, its behaviour such as success or failure has to reflect the implicit security policy; its return values or error status codes can include security-related failures, for example authorisation failure. It may also be required to generate audit records.

Security Selecting Caller

In this case the caller of the primary service is aware of one or more security services and is responsible for managing the security context within which those security services operate. A security selecting caller is capable of operating with several different security contexts, for example, a server application servicing multiple clients, but is not able to authorise any changes to the content of those security contexts.

A security selecting caller can manage the security contexts in two ways:

directly
: The caller directly invokes an infrastructure security service API to configure default values applicable to the primary service API (*out-of-band* with the primary service API). In this case the primary service API could be identical to the security unaware case.

indirectly
: The caller indirectly interacts with the security services by passing the security information as parameters to the primary service APIs (*in-band* with the primary service API). In this case the primary service API specification must include the security information, which may be optional parameters. A primary service provider that supports this type of interaction is termed a *security-supporting* provider.

Security Enforcing Caller

A security enforcing caller is responsible for the enforcement of policy in its use of the primary service providers' services with respect to a particular security service. The primary service provider may have no responsibility for enforcement of policy. A security-non-supporting provider, or a security enforcing provider may surrender responsibility for security policy enforcement to a suitably authorised caller. The caller is responsible for invoking, or implementing, the appropriate security services directly.

4.7 Layering of Security Services

The layering of services is a common characteristic of IT system architectures. For services located at all but the lowest levels within the system, both service functional and data elements are often formed out of the more primitive elements of underlying services through the use of the interfaces to those underlying services. Incorporation of layering of security functionality into IT system architectures is particularly important because:

- it enables a reduction in the amount of trusted code by isolating common critical security functions
- it reduces the exposure of application logic to specific security mechanisms, which are often subject to change.

Figure 4-8 illustrates the concept of the layering of security services for cryptographic-based services.

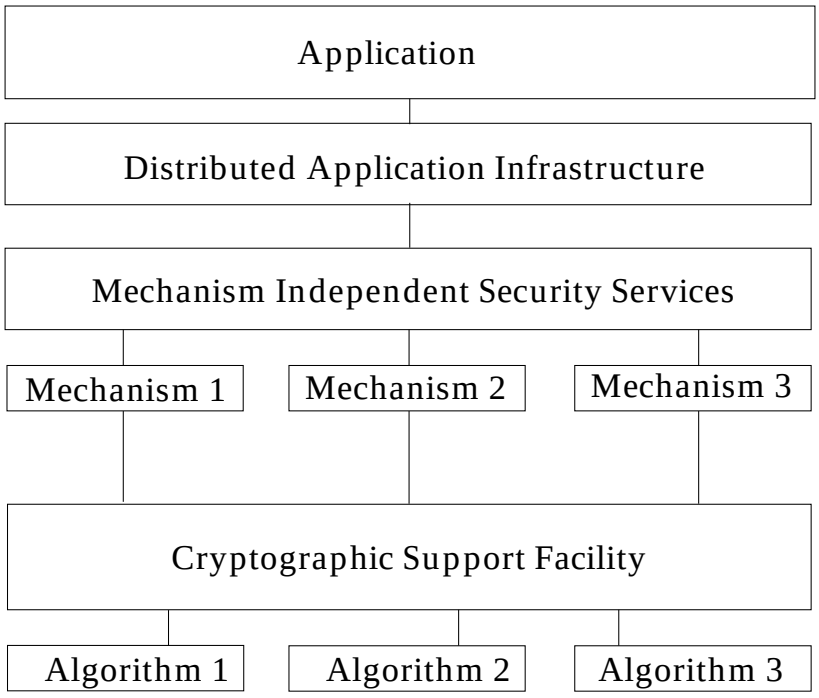

Figure 4-8 Layering of Security Services

At the highest layers are applications that may have no security responsibilities. Security responsibility increases with each lower layer. Conversely the abstraction of security service reduces with each lower layer. The details of the security service become more visible and relevant at the lower layer.

In Figure 4-8 the distributed application infrastructure is responsible for establishing associations but is not responsible for the detail of the establishment of the security context of the association. The infrastructure that establishes the security context is divided into services independent of the key distribution protocol (mechanism) and mechanism-dependent services. These again are distinct from the cryptographic services themselves.

4.8 Trust Boundaries

A further aspect of the structuring of services is the concept of boundaries of trust within the system. This can be expressed in terms of the nature of the security information handled by the service.

Security Unaware Services

These do not handle security attributes or information; the security attributes are enforced below the API. In practice the defaults are normally defined by a security context bound to a tree of execution or a communication channel. Security Unaware services are generally considered untrusted.

Security Selecting Services

These are responsible for managing security attributes by references to a security context handle or as cryptographically protected opaque objects, such as a security certificate. APIs supporting such services therefore include security context handles or opaque data as security parameters. Security Selecting services may be responsible for handling several security contexts simultaneously and selecting the context for a particular operation. If possession of the handle or certificate enables a principal to assert the identity or privileges represented by the security context, a service handling that context must protect such handles or certificates from other, unauthorised, principals.

Security Enforcing Services

These are responsible for directly handling security attributes, such as cryptographic keys in the clear, principal identities and privilege values. The APIs that support such services therefore handle security attributes as specific values. Security Enforcing services are responsible for defining the contents of security contexts and the creation of the security certificates and the use of security attributes extracted from security contexts.

The distinctions between the different types of service represent differences in the required trustworthiness of the services. The distinctions therefore represent trust boundaries that may require enforcement by appropriate separation mechanisms, for example, separate process address spaces or hardware protection rings. Figure 4-9 on page 52 illustrates the concept of the trust status of an application by reference to the nature of the security information it handles.

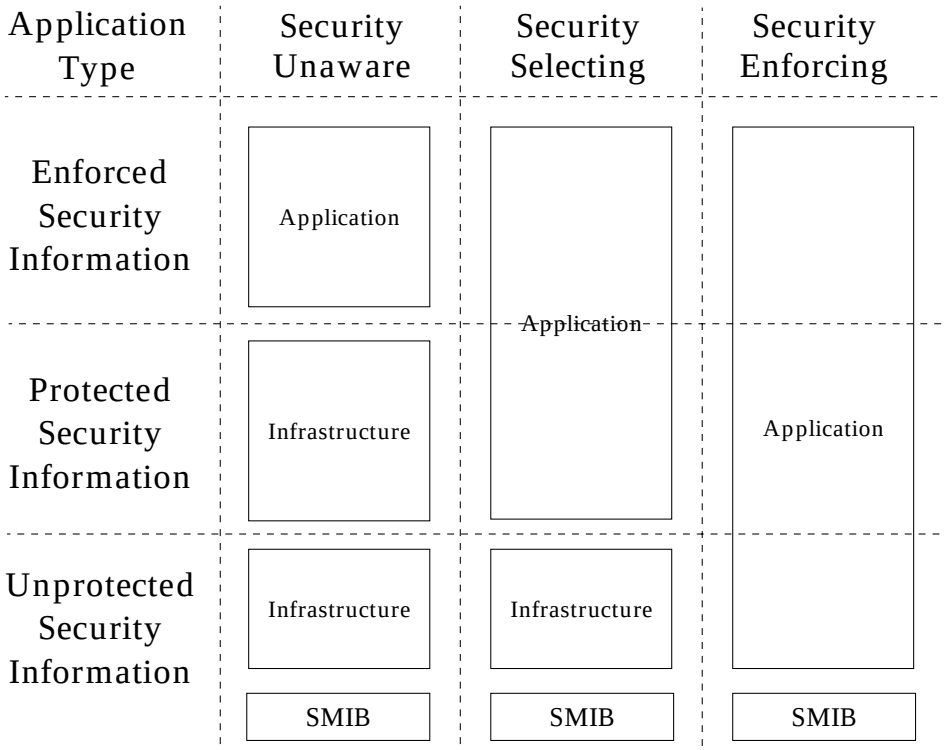

Figure 4-9 Trust Boundaries

Note that there can be trust boundaries associated with each part of the security service and that these trust boundaries are not necessarily coincident. For example, there may be different trust boundaries associated with the handling of label-based authorisation to those associated with ACL-based authorisation. A single level database running on a multi-level system is an example.

Chapter 5

Basic Security Facilities

Chapter 4 introduces a classification of security services and facilities. This chapter describes the services within the basic security facilities as highlighted in Figure 5-1.

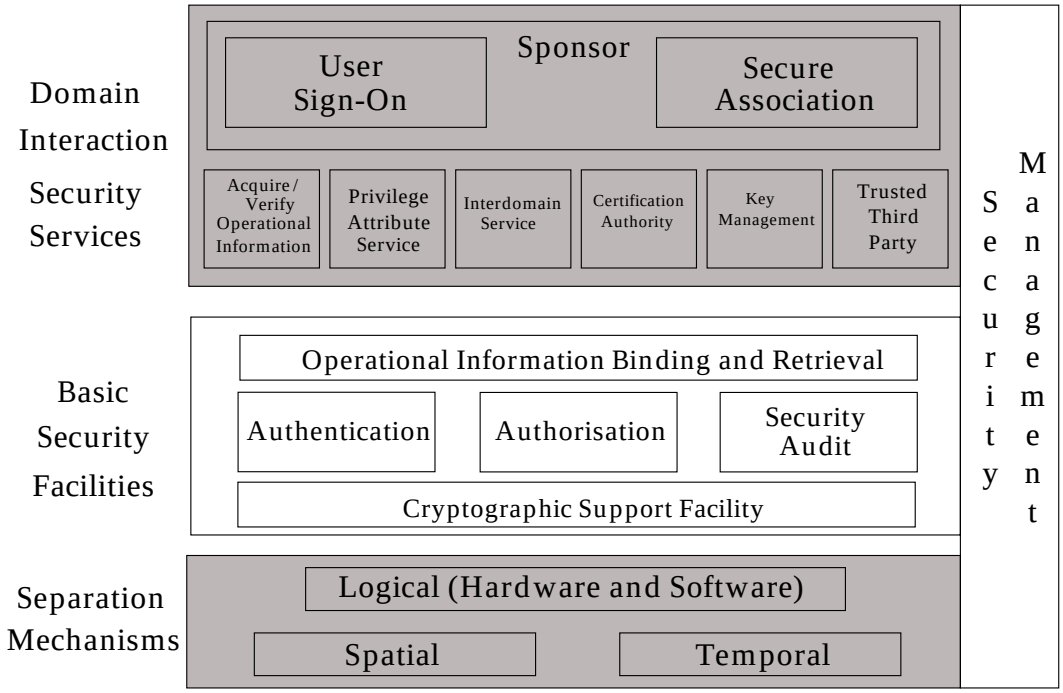

Figure 5-1 Basic Security Services

These services are:

- operational information binding and retrieval
- cryptographic support facility
- authentication
- authorisation
- security audit.

5.1 Operational Information Binding and Retrieval

The security service model described in Chapter 3 refers to the binding and retrieval of security context information for the security service. This requires services that support the following:

Bind Security Context of Principal

This means linking a set of operational security attributes (representing the security context of the principal of the operation) with a security data object, the tree of execution, a data object or the communication channel supporting the operation. For the integrity of the binding to be maintained the security attributes must be securely stored.

Retrieve Security Context of Principal

This is the ability of an authorised component of the tree of execution, a user of a security data object, a data object or a communication channel to obtain the bound security attributes or a handle to them. The attributes or handle would be used directly as part of an operation or to extract component information.

Binding and Retrieval of Data Security Attributes

This is the ability of an application to bind and retrieve security attributes with generated data that represents elements within its domain (for example, to a record in a database or a message) so that the integrity of the association between the data and its security attributes is maintained when that data is stored or exchanged.

Within a platform domain such support is based on the use of the platform domain services. Within and between service domains, the service domains themselves may provide for such binding, possibly using cryptographic based services.

Binding can be an integral part of any of the security facilities described in this document. The full security context of an operation comprises security attributes appropriate to each of the security facilities relevant to that operation.

A principal may operate with multiple security contexts. For example, an application server can service many clients, or a user sponsor can support the concurrent use of several security contexts having different values for some attributes. In this case the principal can select which security context is bound to a specific operation (or set of operations) if this is permitted under the prevailing security policy.

A security context can also be several security contexts representing a compound principal, as can occur with delegation of privileges by one principal to another.

5.1.1 Management Services

There are no management-related aspects of this service.

5.1.2 Operational Services

The operational-related services include Bind-Principal-Security-Attributes and Retrieve-Principal-Security-Attributes.

Bind-Principal-Security-Attributes

This service binds specific operational security attributes (or sets of security attributes), as encapsulated in security contexts, to an entity such as an execution tree, data object, message or a communication channel. For the platform domain, examples of this function are *setuid*( ) and *setgid*( ) in the referenced **XSH**, **Issue 4**, **Version 2** specification. Examples from a distributed service domain are *sec_login_set_context*( ), *rpc_binding_**( ) in the referenced **DCE Security** specification, *gss_acquire_cred*( ) in the **GSS-API Base** specification and *gss_modify_cred*( ) in the **GSS-API Extensions** specification.

A special case of this binding is when a service domain security context is bound to platform security attributes for the purposes of protecting the security context. For example, a privileged remote login daemon may need to make a delegated security context accessible to an unprivileged caller, and would therefore require the permissions on the file containing the security context information to be changed to permit this (see Section 5.1.4 on page 56.)

Retrieve-Principal-Security-Attributes

This service returns the value of one or more selected security attributes or a reference to an encapsulating security context from an entity (an execution tree, data object, message or a communication channel). Examples of security context retrieval from the current execution environment are *sec_login_get_context*( ) in DCE and *gss_acquire_cred*( ) in the **GSS-API Base** specification. Further examples of more specific attribute retrieval are *getuid*( ) in the **XSH**, **Issue 4**, **Version 2** specification, *gss_inquire_cred*( ) in the **GSS-API Base** specification, and *gss_get_attributes*( ) in the **GSS-API Extensions** specification.

Bind-Data-Security-Attributes

This service permits the caller to bind security attributes to a set of data and is typically represented by the calculation of a cryptographic checkvalue over both the data and the security attributes.

Verify-Data-Security-Attributes

This service permits the caller to verify integrity of the binding of security attributes to a set of data and is typically represented by the verification of a cryptographic checkvalue.

5.1.3 Impact on Primary Service APIs

The impact of particular security services on primary service APIs depends on the relative security awareness of the caller of the API.

In general, callers that are security unaware in the context of a particular security service operate within an environment in which another principal has bound a security context to the caller's tree of execution or to the communication channel in use. This means that a default security context has been previously established, in which the security unaware caller operates. An example of this is a user application, such as a word processor, that executes within a user session. The security context is represented by process attributes set as part of the user sign-on procedure.

Callers that are security aware in the context of a particular security service must handle the security attributes relevant to that security service themselves. They can do this by handling the security attributes directly or by using a handle to the security context that contains them. A primary service API supporting a security aware caller may therefore take a security context handle or specific security attributes as parameters.

5.1.4 Implementation

Where the security attributes may be used to assert the authenticated identity of a principal or the principal's privileges, the mechanism supporting the binding must protect against the modification (or if policy so dictates, disclosure) of the security attributes.

Within a platform domain this service is typically supported by storing operational security attributes as process attributes. These process attributes are propagated to new processes by inheritance from the parent process and are protected from access by unauthorised processes. In particular, unauthorised processes are not permitted to modify their own process attributes.

In the case when service domain operational information is added to a system, for which no support is provided by the platform domain, an indirect method of support for operational information binding is used. As an example, the service domain operational information may be held in a file associated with a process attribute, typically a uid. Access to the file is protected by the normal platform domain authorisation services.

Note: This method of binding may not guarantee the uniqueness of the binding of the operational information to a particular execution tree as other execution trees may possess access rights to the file. In addition, any execution tree that can assume responsibility for platform domain access control policy (for example, administrative processes) is able to access such a file.

Ideally platform system services should support the binding of several sets of opaque application-defined operational security information to a thread within a tree of execution.

If operational security attributes are directly handled within user process address space by applications that are not trusted to protect the attributes and their bindings, cryptographic mechanisms must be used to protect the attributes and support the bindings.

5.2 Cryptographic Support Facility

Cryptography supports higher-level security services such as authentication of identities and data-origin, non-repudiation, and data separation and protection. Cryptography is used to provide confidentiality and integrity.

5.2.1 Cryptographic Support Facility Model

Figure 5-2 illustrates a general Cryptographic Support Facility (CSF) which can sit on top of different algorithms and different implementations of those algorithms. The CSF service interface is capable of hiding any specific algorithm, in particular any key format related to the implementation of a chosen algorithm.

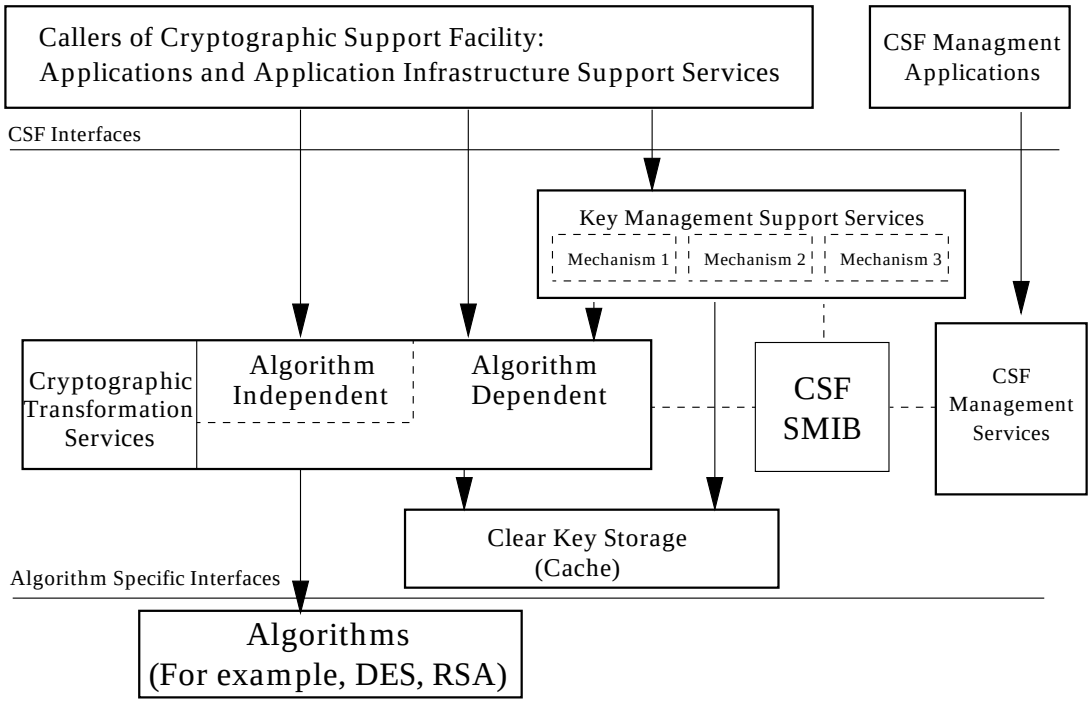

Figure 5-2 Cryptographic Support Facility

The CSF provides support for applications and application infrastructure that:

- need to invoke a given cryptographic transformation or key management operation
- are not concerned about the details of the operation's implementation, nor whether the underlying technology is provided by software or hardware
- may, but need not, specify for a given operation the Quality of Protection needed
- may, but need not, specify for a given operation which particular cryptographic algorithm is used.

The CSF services include provision for data encryption, decryption, production of checkvalues (seals or signatures), checkvalue verification, and the necessary key storage and support services.

CSF services are identified in Section 5.2.4 on page 60 and Section 5.2.5 on page 61.

Note: The area of cryptographic services is evolving and undergoing development. More details of cryptographic services will emerge as implementations mature and efforts to standardise this area are undertaken.

The CSF SMIB may be implemented within the unit that implements the CSF or may be implemented externally to the unit provided it is suitably protected.

5.2.2 Security Considerations

Special controls must be applied to the use of cryptographic software partly due to its fundamental role in distributed system security, but primarily because of legislative constraints on the use and export of software that contains and exposes cryptographic functions. For example, the International Traffic in Arms Regulations (ITAR) impose export constraints on products containing cryptographic services — in particular data confidentiality services. Furthermore some countries impose domestic usage controls.

Hence the CSF interface provider must take into account a number of strict security requirements, which are summarised as follows:

- The CSF must prevent unauthorised access to cryptographic services.
- The CSF must prevent unauthorised access to underlying data such as private or secret keys.
- The CSF must verify any control information associated with keys (such as expiration information) before use.

Figure 5-3 illustrates alternative placements for the ITAR enforcing functions.

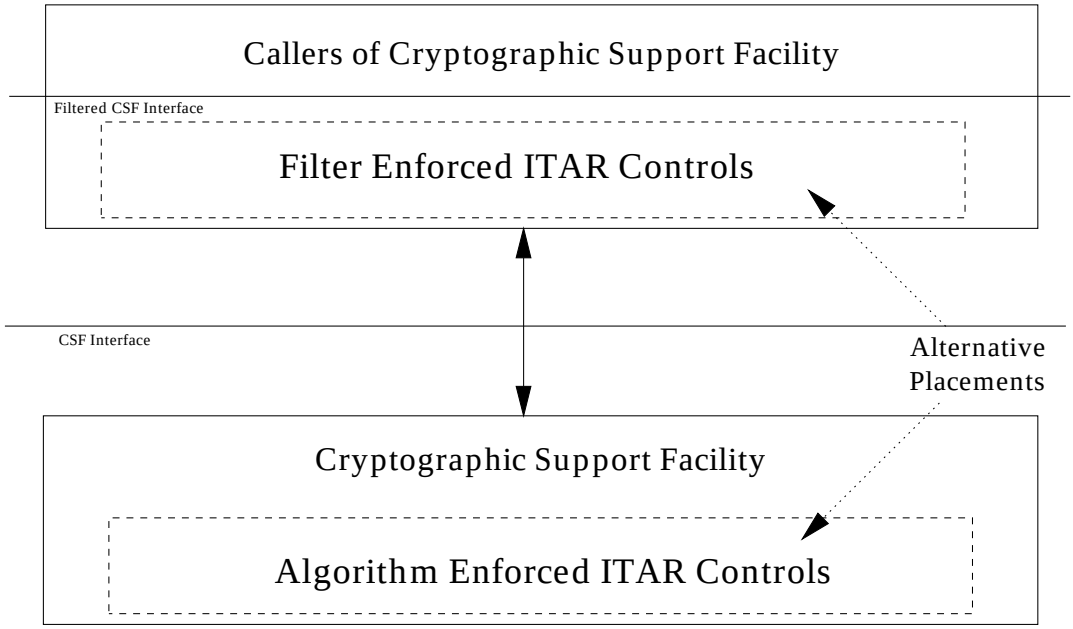

Figure 5-3 ITAR Controls within Cryptographic Support Facility

ITAR enforcing functions may be incorporated in the CSF or within callers of the CSF. If implemented within the CSF, the CSF need not provide the restricted services to callers at all, or it may impose limits on their use. If implemented within the caller of the CSF services, the CSF provides all cryptographic services to its callers, which are then trusted only to utilise the restricted services in an authorised manner. A combination of these alternatives may be deployed, dependent upon one of the following:

- a run-time determination of the caller's authorisation to use the restricted services
- a build-time constraint by restricting the availability of the libraries providing the interfaces to the restricted CSF services to developers of trusted applications.

Some consequences of these requirements are that:

- Conformance to a CSF specification must be compatible with achieving conformance to known domestic and export controls, although different CSF interface profiles may apply for different regulatory environments. For example, a confidentiality service could be an optional service that would not be supported by some conforming implementations.
- The operational CSF services identified in the document are split into:
  - — those that can be used by general applications,
  - — those that can be used by key management applications that are only authorised to handle encrypted keys, and
  - — those that can be used by key management applications that are authorised to handle clear keys.
- The management CSF services identified in the document are split into:
  - — key activation and deactivation services
  - — Master-Key services
  - — algorithm-management services.

  Note: Additional functions to those detailed in this framework can be identified but this is left to subsequent, more detailed work.
- For those CSF services identified to be ITAR-sensitive it should be possible to achieve compliance with ITAR or similar rules through internal CSF controls (for example, by binding usage controls into algorithms), or through controls at the CSF interface layer (for example, an *ITAR filter* as shown in Figure 5-3 on page 58).
- Depending on the policy enforced, the CSF might require its callers to have been authenticated before they can access its services. A cryptographic product can therefore include authentication and authorisation services, as well as the management and operational cryptographic services.
- Above the CSF interface operational keys should never be stored or handled in the clear by unauthorised callers. For handling by such callers, operational keys are protected above the CSF interface by enciphering with the CSF Master-Key, a Key-Encrypting-Key or an Archive-Key. Authorised callers are trusted key distribution services that require to combine an operational key in the clear with other related information to create a mechanism-specific token. Also note that subversion of CSF access controls is more important for services related to key management than those related to applications.

5.2.3 Technical Constraints

As cryptographic interfaces are often implemented in hardware, the CSF interfaces and constructs identified in this framework should not require implementations to maintain internal state information across API invocations.

The CSF services could be achieved by means of stateless transactions in which state information is provided as parameters (either by value or by reference) of the current API invocation, and not based on information retained from previous API invocations.

5.2.4 Management Services

5.2.4.1 Algorithm Management

These services provide for the management of cryptographic algorithms.

Install-Algorithm

This service installs an encryption algorithm for use within a system. This can include configuration of its uses. For example, some algorithms may be restricted to use for key distribution and digital signature purposes only. These constraints may be imposed by a system administrator for reasons of performance or local security policy. Alternatively, they may be imposed by a vendor or integrator at system build time because of international trade restrictions or national security policy.

Deinstall-Algorithm

This service removes an algorithm from a system.

Note: Although removed from current use within the system the algorithm may be needed to support services such as audit analysis and non-repudiation on historic data. For this reason the deinstallation of an algorithm need not include its complete deletion from the system.

Disable-Algorithm

This service disables an algorithm and thus inhibits its use.

Enable-Algorithm

This service enables an algorithm previously disabled.

Set-Defaults

This service maps Quality-Of-Protection (QOP) values to algorithms and algorithm parameters.

5.2.4.2 Master-Key Services

One of the security considerations discussed in Section 5.2.2 on page 58 is the necessity for the CSF to prevent unauthorised access to Clear-Keys when they are passed outside the CSF boundary (that is, outside the TCB boundary.) This is supported by the concept of a Master-Key. This is a key, specific to the CSF which is used by the CSF to encipher any CSF data (in particular keys) passed to callers of the CSF that are not authorised to handle Clear-Keys. The set of management services are related to the handling of Master-Keys are:

Load-New-Master-Key

This service loads a new master key into the CSF. The Master-Key may be loaded as several parts, each loaded by a separate user.

Set-Current-Master-Key

This service changes the key being used as the current Master-Key by the CSF.

Re-encipher-to-New-Master-Key

This service changes the Master-Key under which keys are stored from a previous Master-Key to the New Master-Key. This operation is used as part of the procedure for changing the CSF Master-Key to change the Master-Key used to encipher all current keys.

Re-encipher-to-Current-Master-Key

This service changes the Master-Key under which a key is stored to the current Master-Key. This service may be used, for example, to re-encipher keys retrieved from archive.

5.2.4.3 Key Activation Services

A cryptographic key may be active, available for operational use, or inactive, not available for operational use. This state change may occur by a time trigger, based on a key validity period, or may be specifically caused by the following services.

Note: The key state is generally more complex than just active and inactive. however, this is an unnecessary detail at the level of this framework and is left for subsequent work to detail.

Activate-Key

This service activates or reactivates an inactive key. For example, a key may need to be reactivated on retrieval from an archive in order to validate a digital signature.

Deactivate-Key

This service deactivates an active key. For example, this may be invoked because a key has been compromised.

Combine-Key-Parts

This service supports the re-creation of a key from a combination of separate key parts.

5.2.5 Operational Services Overview

As discussed above, the operational CSF services are structured into four categories:

- those that can be used by general applications
- those which are ITAR-sensitive
- those that can be used by key management applications that are only authorised to handle encrypted keys — these types of caller can also use the general application services

- those that can be used by key management applications authorised to handle clear keys — these types of caller can also use the general application and encrypted key management services.

5.2.5.1 General Application

These services allow callers to apply, and where appropriate, reverse cryptographic transforms on data. They may be used by applications or by security service providers, for example authentication and access control information service providers for the generation of security certificates or security tokens.

Create-Cryptographic-Context

This service enables callers to create a cryptographic context by selecting from a range of options for subsequent cryptographic transformations. These options may, subject to policy, include choosing an administratively-defined quality of protection, cryptographic algorithms, modes of operation or other parameters specific to the cryptographic algorithms in use.

The cryptographic context created is used in subsequent calls to cryptographic transforms. References to such contexts should be as an opaque parameter (which may be an integer handle or a cryptographically protected token).

Release-Cryptographic-Context

This service releases, or deletes, the Cryptographic-Context referenced by the specified handle.

One-Way-Function

This service irreversibly digests the input data using a given algorithm and related parameters specified by means of a cryptographic context, in which case it is also known as a *Hash* function.

Generate-Time-Variant-Parameter

This service emits one of the following time-variant parameters:

- time
- high quality random number
- nonces
- sequence numbers.

Generate-Check-Value

This service returns the seal of the input data computed using the cryptographic algorithms, key and related parameters as specified directly or by means of a context. This service applies a checkvalue to a data item prior to, or as part of data transfer for the purposes of integrity, data-origin authentication, or password encryption.

Verify-Check-Value

This service validates the integrity of the content of a set of data, and authenticates the origin of a set of data against the supplied checkvalue previously created by Generate-Check-Value. This service checks the integrity of the input data using the given cryptographic algorithms, key and related parameters as specified directly or by means of a context. The seal of the input data is computed and is then compared with the given proof.

Check-Value-Translate

This service translates a checkvalue from one cryptographic context to another. It effectively comprises a Verify-Check-Value operation followed by a Generate-Check-Value. It could be used, for example, by a trusted third party acting as an intermediary between two other principals.

5.2.5.2 ITAR-sensitive Services

The following services, which provide for the enciphering of application data, are ITAR-sensitive.

Data-Encrypt

This service returns the input data encrypted under a given reversible cryptographic algorithm, key and related parameters specified directly or by means of a context.

Data-Decrypt

This service returns the input data decrypted using the given reversible cryptographic algorithm, key and related parameters specified directly or by means of a context.

Ciphertext-Translate

This service translates ciphertext from one cryptographic context to another. It effectively comprises a Data-Decrypt operation followed by a Data-Encrypt operation. It could be used, for example, by a Trusted Third Party acting as an intermediary between two other principals.

Protect

This service enables callers safely to invoke a combination of cryptographic services, such as those described above, to achieve data origin authentication, data integrity, data confidentiality, and so on. Combining services to achieve all objectives is open to error, for example, the method used to archive data integrity might leak information and so comprise data confidentiality.

Unprotect

This service enables callers to verify the authenticity, integrity, confidentiality, and so on of a data object protected by the Protect service.

The primary rationale for the Protect and Unprotect services is to enable callers safely to invoke a combination of cryptographic services, such as those described above, to achieve data origin authentication, data integrity, data confidentiality, and so on. Combining services to achieve all objectives is open to error. For example, the method used to achieve data integrity might leak information and so compromise data confidentiality.

Additional rationales are to allow:

- simplification for the caller by allowing some details to be controlled below the API

- potential performance improvement over multiple calls to the CSF.

The protection is applied to, or verified for, the input data using cryptographic algorithms, keys, and related parameters as specified directly or by means of a cryptographic context.

5.2.5.3 Encrypted Key Management Services

These services are used by key management applications that handle encrypted keys. The keys may be enciphered under the CSF Master-Key, a Key-Encrypting-Key for export or import, or an Archive-Key for archive purposes.

Derive-Key

This service derives a secret key from a parameter supplied to it. The parameter may be a text string, a bit string or a key reference.

Export-Key

This service exports a key and associated information in a protected form using a Key-Encrypting-Key. This service may return an encrypted key or a mechanism-specific token, including the key, depending upon the specified input parameters. This service is provided to support key distribution services.

Import-Key

This service imports a key and associated information in a protected form as produced by the Export-Key service. This service is provided to support key distribution services.

Generate-Key

This service generates a secret key or public and private key pair enciphered under the CSF Master-Key. The algorithm, key-size, usage, lifetime and other associated parameters are specified by a cryptographic-context.

Archive-Key

This services transforms the operational key, ciphered with the current master-key, into a key ciphered with an archive key and stores it in a long term storage unit.

Restore-Key

This service retrieves a key ciphered with an archive key out of the long term storage unit and transforms it into an operational key ciphered with the current valid master key.

Delete-Key

This service deletes a key from the CSF.

Key-Test-Generate

This service generates a test pattern for the specified key.

Key-Test-Verify

This service verifies a key against the specified key test pattern.

The purpose of the key test services are to test the equivalence of two supposedly identical keys generated in different locations or using two different implementations.

5.2.5.4 Encrypted Key Management Services

These services are used by key management applications that handle keys in the clear. This is particularly the case with current implementations in which CSF services do not yet provide mechanism-specific support for Key-Export and Key-Import operations.

Generate-Clear-Key and Generate-Clear-Key-Pair

This service generates a secret key or public and private key pair in the clear. The algorithm, key-size, usage, lifetime and other associated parameters are specified by a cryptographic context.

Load-Key

This service loads a Clear-Key into the CSF.

Key-Encrypt

This service is used to encipher a key and key-related data. It is distinguished from Data-Encrypt, for example, by constraints on the size of data that may be enciphered, or the speed at which it may be enciphered.

Key-Decrypt

This service is used to decipher a key and key-related data. It is distinguished from Data-decrypt, for example, by placing constraints on the size of data that may be deciphered, or the speed at which it may be deciphered.

5.2.6 Impact on Primary Service APIs

A caller of a primary service API that is unaware of the cryptographic service is illustrated in Figure 5-4 on page 66.

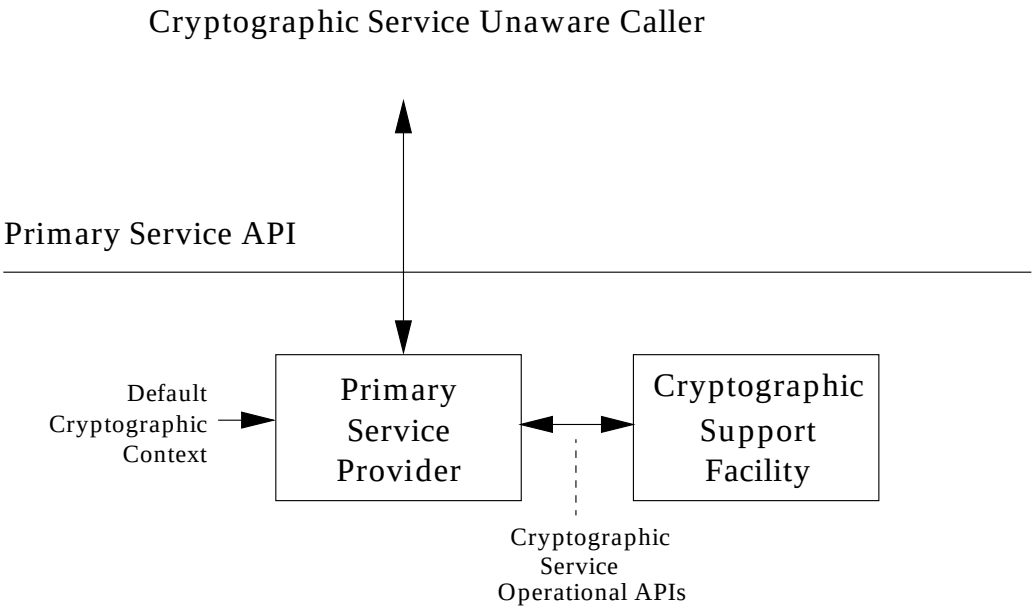

Figure 5-4 Cryptographic Service Unaware Caller

The caller of the primary service API is unaware of, and not responsible for selecting, the cryptographic context used. The primary service provides no support for the caller to specify the cryptographic context used by means of the primary service API.

The cryptographic context is taken as a default previously established and associated with the CSF itself or the particular tree of execution making the call.

A caller of a primary service API that is aware of the cryptographic quality of protection (QOP) is illustrated in Figure 5-5 on page 67.

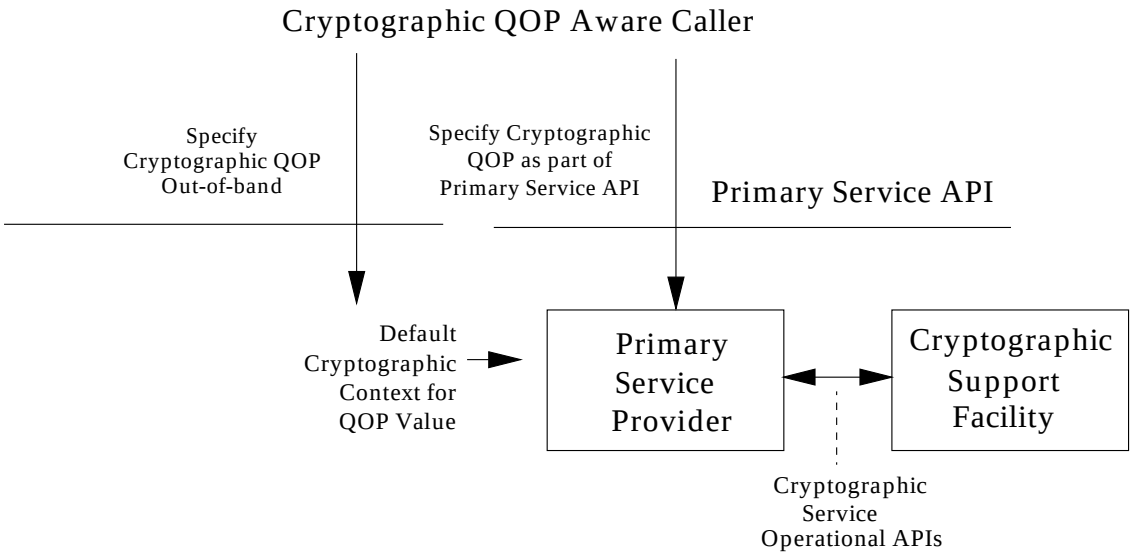

Figure 5-5 Cryptographic QOP Aware Caller

The caller of the primary service API is aware of, and responsible for selecting, the QOP to be used. The primary service provider can be security supporting, providing support for the QOP to be specified as part of the primary service API. Alternatively, the caller must specify the QOP in an out-of-band manner by setting the default. This default is applied to any cryptographic service calls arising from invocation of the primary service API for the particular tree of execution. The specific algorithm used is that configured as the default for that QOP within the CSF SMIB. A caller of a primary service API that is aware of the cryptographic algorithm is illustrated in Figure 5-6.

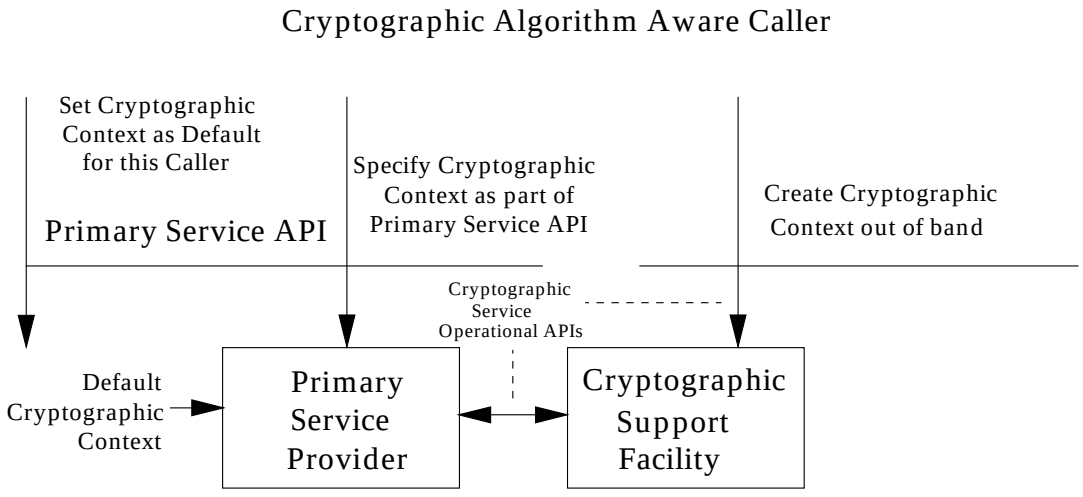

Figure 5-6 Cryptographic Algorithm Aware Caller

The caller of the primary service API is aware of, and responsible for selecting, the cryptographic algorithm used. The primary service provider can be security supporting, providing support for the cryptographic context to be specified as part of the primary service API. Alternatively, the caller must specify the cryptographic context in an out-of-band manner by setting the default cryptographic context. The default cryptographic context is applied to any cryptographic service calls arising from invocation of the primary service API for the particular tree of execution.

The callers of the CSF may themselves be structured. This is illustrated in Figure 5-7.

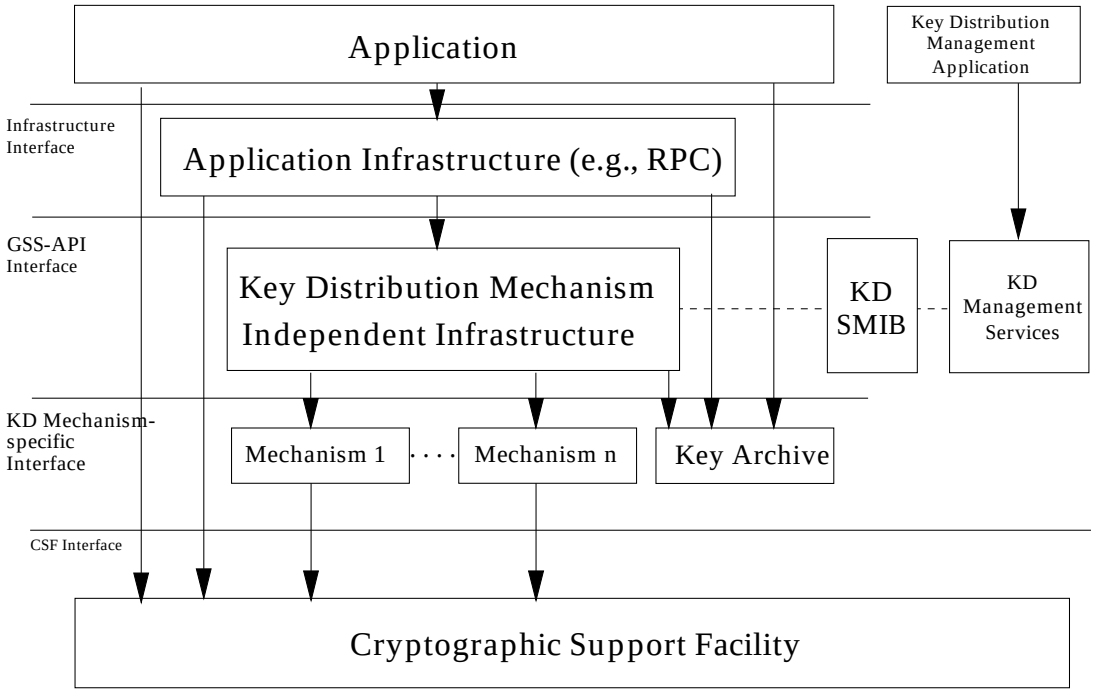

Figure 5-7 Structure of CSF Callers

The trust boundaries represented by the different layers of interface are of particular significance. As discussed in Section 5.2.2 on page 58, the CSF interface represents a boundary above which cryptographic keys are not stored or manipulated in the clear by unauthorised callers. Above the CSF interface, keys are normally referenced by a handle or are handled as opaque data cryptographically protected according to import, export and archive key management transformations. Applications that are authorised to handle cryptographic keys in the clear constitute part of the Cryptographic Service TCB; they require commensurate protection and segregation from components that are not part of the Cryptographic Service TCB.

5.3 Authentication

Authentication services corroborate the claimed identity of a principal. A principal's authenticated identity is used by other security services and operations. Consequently, the deployment and use of strong identification and authentication services and mechanisms is fundamental to the success of the overall security solution. To counter unauthorised use through masquerade, these services and mechanisms are required at points of access to the information systems and the services they provide.

The basic authentication strategy is to use either secret or unforgeable and unique information shared by the requesting principal and the identification and authentication mechanisms. Different authentication techniques exist in existing practice and emerging standards; the techniques involve different mechanisms for proving identity. Examples include passwords, secret information held on tokens or smart cards, and biometric information (for example finger prints or retinal scans) unique to each principal. The primary mechanism used to protect against unauthorised disclosure of a principal's secret credentials is cryptography.

In general there are two forms of authentication:

- principal authentication
- data origin authentication.

Principal authentication provides corroboration of the identity of a principal, while data origin authentication provides corroboration of the identity of the principal responsible for the creation of a specific data item.

5.3.1 Model

Figure 5-8 illustrates the basic authentication model. This incorporates some concepts from the ISO model as described in ISO / IEC 10181-2.

Note: In the ISO / IEC 10181-2 *claimant* is equivalent to *initiator*; *verifier* is equivalent to *target*.

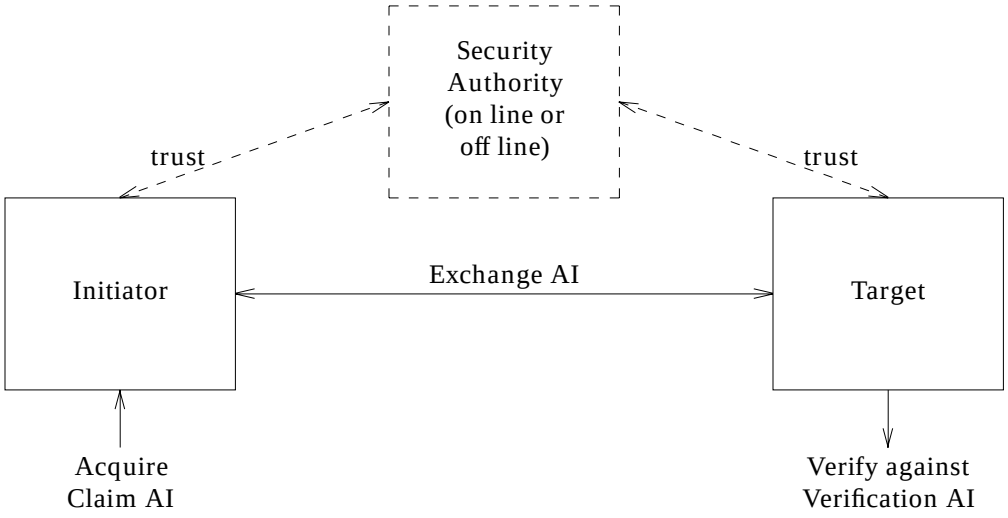

Figure 5-8 Basic Authentication Service Model

The initiator represents a principal for the purposes of authentication. The target verifies the authenticated identity of the initiator.

The security authority that is trusted by the other principals with respect to security-related operations.

AI means authentication information. *Claim AI* is information needed to initiate the authentication process on behalf of the principal, for example, a password, a secret key or a private key.

Verification AI is used to verify an identity claimed through exchange AI, for example *password* related to identity of principal, *secret key* related to identity of principal or authority, *public key* related to identity of principal or authority.

Exchange AI is information exchanged between an initiator and a target during the process of authenticating a principal. Examples are: *claimed distinguishing identifier*, password, challenge, response to challenge, test number, verifier identifier, result of transformation function, on-line certificate, off-line certificate.

Note: See the ISO/IEC 10181-2 for further detail on variations of this model: No Third Party, In-line Authentication, On-line Authentication and Off-line Authentication.

Data Origin Authentication

Data origin authentication generally relies upon the signing or sealing of a data item by the originator and the corresponding verification of the signature or seal by the recipient.

This service is the basis of the verification of security certificates and tokens on which distributed data authentication and distributed privilege attribute services are based.

5.3.2 Local User Authentication

Figure 5-9 on page 71 illustrates authentication in the user sign-on procedure.

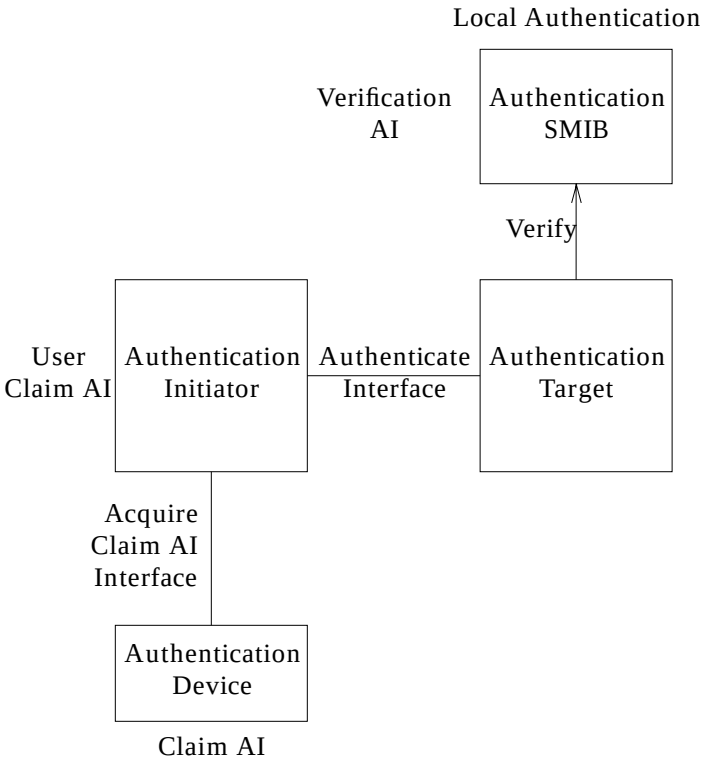

Figure 5-9 User Sign-on Authentication

The authentication initiator is the means by which the user interacts with the system. The authentication initiator acquires Claim AI by interrogation of the user, by reading information from an authentication device, or both. Examples of authentication devices are badge readers, smart card readers and biometric devices.

The Claim AI may be preprocessed by the authentication initiator (for example, encrypted) and submitted as Exchange AI to the authentication target. The target verifies the Exchange AI against Verification AI held in the authentication SMIB, for example, a one-way encrypted password.

The authentication target returns an authenticated identity to the authentication initiator which can then be used to assert the user's identity as a principal within the system when requesting services.

The authentication service can be based entirely within a platform and reference a local authentication SMIB. Standalone system authentication, such as traditional login, is an example in which an authenticated identity particular to the platform is returned.

Alternatively, the authentication service may invoke a distributed authentication target which returns an authenticated identity of more universal applicability within the security domain serviced by the authentication target. An example of such a service is DCE login.

In the distributed authentication service case the authenticated identity may be in the form of a security certificate signed by the authentication service.

5.3.3 Application Authentication

Figure 5-10 illustrates authentication of an application as a principal.

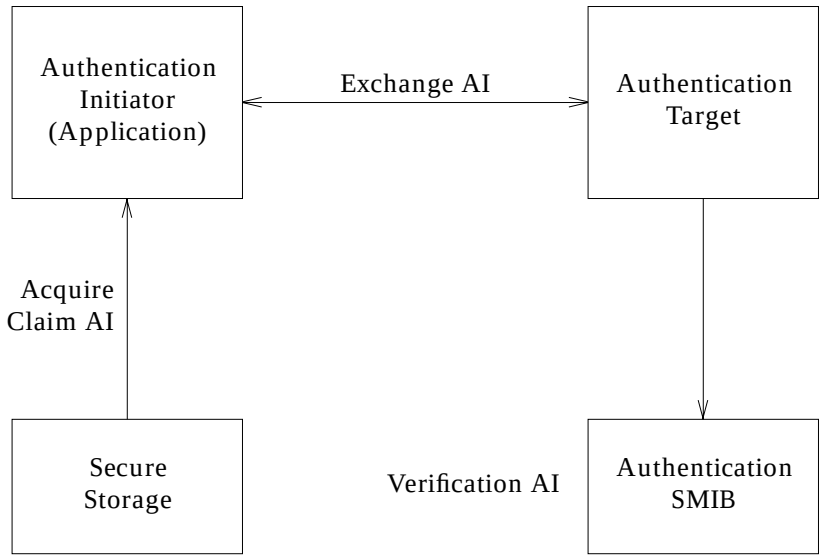

Figure 5-10 Application Authentication

In this case the application retrieves the Claim AI (such as a cryptographic key) from secure storage and submits this as Exchange AI to the authentication target. The authentication target checks the Exchange AI against Verification AI (such as the secret key shared with the application, or the public key of the application if asymmetric cryptography is used).

This style of authentication may be used to support user authentication when the authentication initiator is required to authenticate itself to the authentication target, or the authentication initiator requires the authentication target to authenticate itself to the authentication initiator, or both.

The strength of this form of authentication depends upon the secure storage of the application Claim AI and its binding to the application. This depends upon the security services of the platform supporting the application. The initialisation of the Application AI in secure storage is a management procedure associated with application installation and initialisation. Procedures could be required to change the Application AI periodically as required by the security policy.

5.3.4 Secure Association Authentication

Figure 5-11 on page 73 illustrates authentication as part of the Secure Association Service.

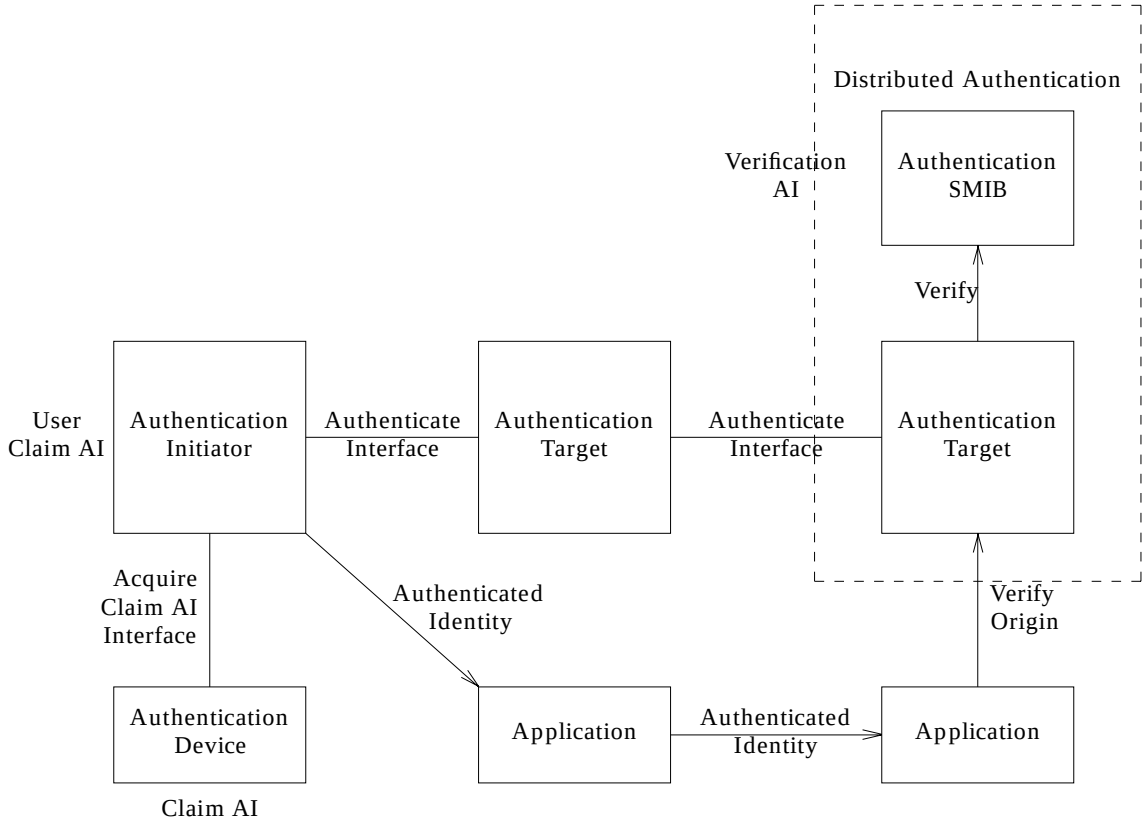

Figure 5-11 Secure Association Authentication

In this case the user authenticates as part of the User Sign-on procedure and the authentication target returns an authenticated identity as a security certificate to the authentication initiator. This authenticated identity is used by the application invoked by the authentication initiator to assert the user's identity to the target application. The target application may verify the authenticated identity by verifying the signature on the security certificate (see Section 6.3 on page 110).

5.3.5 Authentication Device Interfaces

The five principles on which authentication can be based are:

- something known, for example, a password
- something possessed, for example, a smart card, hand-held device or machine-readable badge
- some permanent characteristics, for example, fingerprints, signature or retinal scan
- accepting that a trusted third party has established authentication
- context, for example, the address of the principal.

Existing products on the market provide some or all of these options through a variety of incompatible interfaces. To bridge the gap between the authentication model in the preceding section and existing practice, interface classes for different types of devices must be identified,

and the relationship between those interfaces and the higher-level authentication interfaces in the model need to be shown. In particular, these interfaces must enable user authentication initiators and applications to function with different authentication systems and be vendor independent.

It must be possible:

- to enable different classes of authentication devices such as:
  - — simple magnetic-stripe cards
  - — password-protected cards containing a user's secret key
  - — smart cards containing a principal's private key
- to support different (both software and hardware mechanisms) for implementation of non-disclosing passwords
- for the system to be extensible to support devices with authorisation information (such as long-term authentication and access control certificates)
- to facilitate authentication devices that interrogate the user, either directly or by means of a system-supplied Human-Computer Interface (HCI).
- to set device indicators (if any) correctly.

Figure 5-12 on page 75 shows the different classes of authentication devices and the location of the interfaces to them.

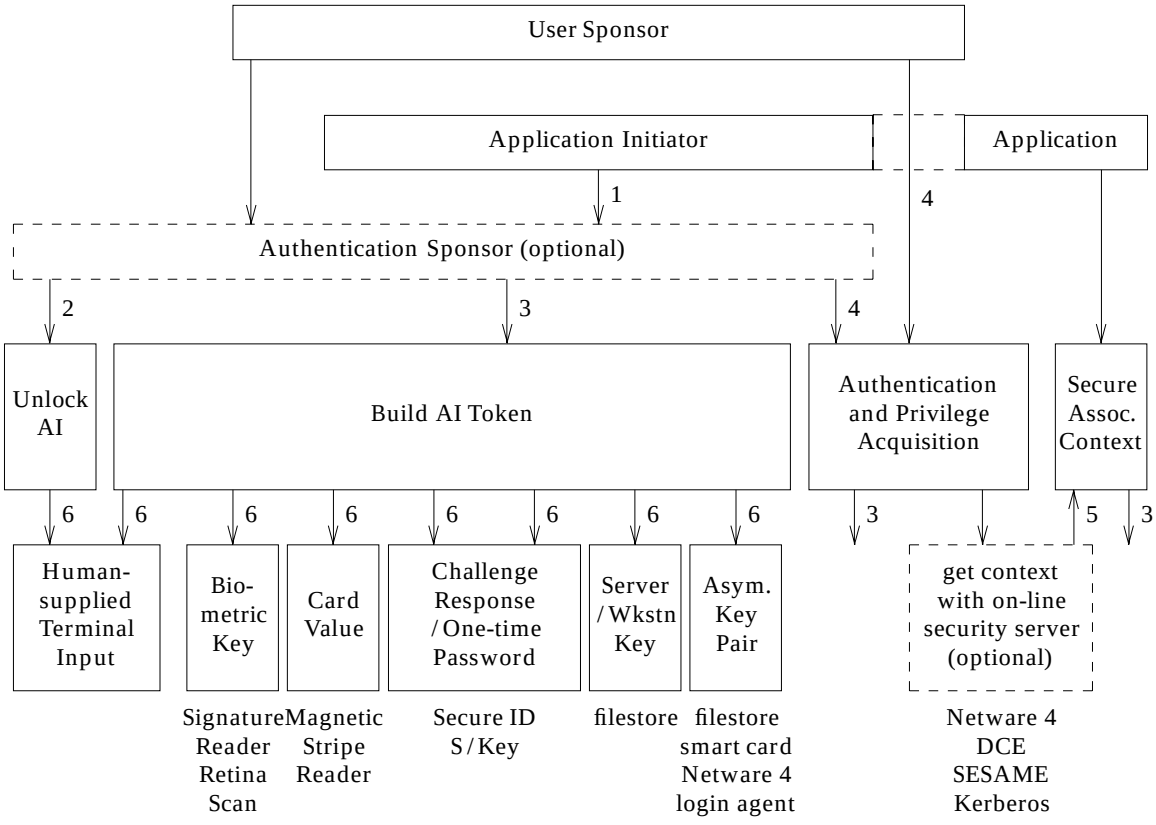

Figure 5-12 Authentication Device Interface Model

The interfaces illustrated are as follows:

1. **Acquire Authentication Information**

   This interface returns the required Exchange AI for subsequent authentication to an on-line security server or a communicating peer. The *authentication sponsor* is the logic that is aware of the specific class of authentication mechanism or mechanisms required, and whether they need to be unlocked before use. In this model, it is optional whether User Sponsors and Applications invoke interface 1, or alternatively interface 2, 3 or 4. The advantage of interface 1 is that mechanism-specific authentication sponsor code does not have to be embedded into each User Sponsor and application.

2. **Unlock AI**

   This interface obtains any information from the user for the purpose of releasing Claim AI (for example, for the case where a smart card requires a user (or potentially more than one user) to authenticate to it, or when a user's private key is held encrypted).

3. **Build AI Token**

   This interface obtains the Claim AI from the required device or devices, and assembles it into a token.

4. **Authenticate to On-line Security Server**

   Where an on-line security server is used, this interface passes Exchange AI to obtain a security certificate for use in subsequent secure associations.

5. **Deposit Security Information**

   This interface deposits security tokens or certificates for use by the secure association service in subsequent authentication.

6. **Get Claim AI**

   This interface obtains the Claim AI from the given authentication device. Each different class of device in turn requires a standardised interface.

A further interface (not illustrated) is required in the component which verifies authentication information. This interface would verify the received AI token.

The following sections outline potential scenarios for the provision of authentication interfaces. They describe interfaces that have been proven through implementation in products.

Magnetic Stripe Reader

Workstations can be optionally equipped with a magnetic-stripe reader for use during authentication. Such readers would typically enable a card to be inserted as part of a user-to-host authentication operation, and would support asynchronous notification to the host of card insertion and removal. Hardware card-reader indicators are sometimes used. if present, they are usually under software control. In such cases, the following interfaces should be supported:

- initialise or reset card reader
- poll for card insertion
- read card data
- poll for card removal
- set card reader display.

The format for the card data should be defined such that a consistent interpretation can be made of different vendors' authentication information and expiry information, because standards are defined to support many different options. Management interfaces should be provided such that a card can be read by an administrative program and associated with the user.

One-time Password Devices

To support *non-disclosing* passwords, there is typically a hardware device which provides the next password. Sometimes this is software-based or even occasionally on paper.

Users are generally expected to read the password and type it in as normal. Client software interfaces at the user's workstation are not necessarily required to integrate one-time password devices into an existing distributed authentication system. However, there is a need to integrate the new password checking logic at the server side.

For vendor-independent integration of different one-time password devices to be possible, a generic interface is required that can potentially be linked with an authentication server and support the server-side *password checker*. The interface must support:

- verification of passwords
- re-synchronisation of device and password checker.

Management interfaces should be supported, where required, to initialise the client and server side.

Password-protected Authentication Device

Authentication devices can be provided that maintain their own authentication database of passwords required for user sign-on to different systems. Examples of such devices are security products for PCs or authentication tokens which are carried around by users. When integrated into a distributed environment, the implementation of single sign-on demands that it should be possible for a distributed authentication system (such as DCE) to work with password-protected devices. This needs a policy decision as to which system should be on top (that is, make final decision on user access for the system). Solutions based on workstations or distributed systems are possible.

For the workstation-based solution, the workstation authenticates the user; if successful, the distributed system authentication takes place. The workstation login is not dependent on the distributed system login.

Note: For single login, this approach requires synchronisation of user details on workstations and in the distributed system. It also requires an interface between the workstation and the distributed system so that authentication information can be obtained.

For the solution based on a distributed system, the user authenticates to the distributed system using authentication information supplemented by the workstation. This is similar to the model of the workstation as an ECMA *Primary Principal* (see the referenced ECMA document).

Smart Card Authentication Device

The general principal of smart card authentication devices is that they are made of a chip that stores a key (the smart card secret) and implements a cryptographic algorithm. The chip can only be activated by a secret number known to the smart card owner. The secret number could be a PIN (as is found in today's versions of smart cards) or could be the result of more sophisticated processing. Examples of such processing are:

- a biometrics analysis of a finger print
- a voice recognition system
- a password generation or verification system performing functions related to password ageing, strength, and so on (see the **Procurement Guide**).

  Authentication of a principal with a smart card is performed in two steps:

  1. The principal authenticates itself to the smart card by submitting a secret number, the Personal Identification Number (PIN), to the smart card. If the PIN is correct, the chip on the smart card is activated.
  2. A trusted party that knows the smart card secret, authenticates the smart card, by challenging the smart card with a random number, and then by checking the certificate computed by the algorithm on the smart card from the challenge and the smart card secret.

Three basic interfaces are needed to handle a smart card device:

- an interface to submit the PIN and activate the chip of the smart card
- an interface to compute a certificate on the chip from a challenge and the smart card secret
- an interface to deactivate the chip.

Biometrics

A description of the technology and interfaces will be included in a future release of this document.

5.3.6 Management Services

Authentication management services include distribution of descriptive information, passwords or keys, to principals required to perform authentication. Authentication management services include Install, Deinstall, Change-AI, Distribute, Disable and Enable.

Install-AI

Installs an account, including verification AI, into the authentication service SMIB. The install operation can be a compound operation comprising:

enroll
This adds authentication information associated with a principal to the authentication service SMIB.

testify
This provides additional information by a witness principal. An example of this is an *n*-person rule operation or split knowledge operation.

validate
This introduces a principal into a security domain (activates a principal record) and is performed on behalf of a security authority. It may also be used to revalidate a principal

previously invalidated.

confirm
This is invoked after a validate service and returns specific information such as acknowledgement or rejection of enrollment to a principal. This may be performed out of band, for example by physical means.

Deinstall-AI

Deinstall-AI removes an account. That is, it causes a principal to be removed from a group of (possibly authenticated) principals. Deinstall can be qualified with:

invalidate
This causes a principal to be disabled from authenticating, but leaves the account record present in SMIB. This is also termed *locking out an account*.

notify
This can be invoked by the security authority after an invalidate service to notify the principal of its invalidation and possibly information on how to re-enroll.

unenroll
This causes a principal to be suppressed as a member of the population of principals of a security domain that may be authenticated.

Note: The term suppress is used in preference to delete as the authentication information associated with the principal may have to be retained to support other security services such as audit analysis based on historic records.

Change-AI

Change-AI is invoked on behalf of a principal or security authority to cause AI to change, for example, change password.

A further purpose of Change-AI is the modification of AI to control the requirement for the use of additional or alternative means of authentication, for example, the use of authentication devices such as smart cards or badge readers.

Generate-AI

Generate-AI produces one or more items of authentication information which meet security policy for selection by the user.

Distribute-AI

Distribute-AI is a service that enables any principal to acquire sufficient Verification AI upon which to verify Exchange AI. An example is the distribution of cryptographic keys to verify cryptographic seals or signatures applied to authentication certificates.

Disable-AI and Enable-AI

Disable-AI causes a state to be established whereby a principal is temporarily unable to authenticate. Enable-AI causes the disabled state to be terminated thus re-enabling the principal to be authenticated.

Note: Disable-AI and Enable-AI appear similar in effect to invalidate and validate. However, validate implies an aspect of authorisation is associated as a requirement with the exercise of the service whereas enable does not.

Map-Text-String-to-AI

This function maps a text string to the internal form of AI, for example, mapping a principal name to an identity. This function is used by management applications that support the installation and modification of AI. It is also required by audit analysis services, possibly with respect to historical AI.

Map-AI-to-Text-String

This function maps an internal form of AI to a text name, for example, mapping an identity to a principal name. This function is used by management applications that support the listing and modification of AI. It is also required by audit analysis services, possibly with respect to historical AI.

5.3.7 Operational Services

Authentication operational services are those services invoked as part of the actual authentication process. Authentication operational services include acquire, generate and verify.

Acquire-AI

Acquire-AI allows a sponsor or verifier to obtain Claim AI required to generate Exchange AI for an instance of authentication. This may require interacting with an authentication device, for example, a smart card or other device (see Section 5.3.5 on page 73 for a more detailed view of this interface).

Verify-AI

Verify-AI is invoked by a verifier to verify Exchange AI received from the sponsor, or to create Exchange AI for transfer back to the claimant. This includes the verification of Exchange AI associated with data for data origin authentication.

Authenticate-and-Acquire-Privileges

Authenticate and acquire privileges from an on-line security server.

Terminate-Authentication-Context

Terminate context with on-line security server.

5.3.8 Impact on Primary Service APIs

Within an IT system the need to authenticate a principal's identity arises in three specific circumstances:

- the identification and authentication of a user, physically distinct from the IT system
- as part of the creation of a secure association between security domains within an IT system, or between IT systems
- data origin authentication.

User authentication is the responsibility of the components providing the User Sign-on function. User authentication within distributed systems does not need to be invoked by other applications as the distributed security services provide alternative methods (that is, Secure Association Service).

Because distributed security services are not available in all networks, applications that support interconnection (such as ftp and telnet) may have to support user authentication and various authentication methods. For this purpose it could be appropriate to specify an API to user authentication services.

Data origin authentication is supported by the cryptographic services of signing and sealing data (Generate-Check-Value), and the corresponding services of unseal and verify signature (Verify-Check-Value). These services are defined in **Generate-Check-Value** on page 62 and **Verify-Check-Value** on page 63 respectively.

5.4 Authorisation

Authorisation services are used to counter unauthorised modification and disclosure threats.

A range of authorisation and authentication services can be combined to form an access control regime. The general purpose is to ensure that only those principals (known as initiators) with the necessary authorisations are allowed to access information and information system resources and services (known as targets).

Authorisation services may be used at many places within a system. Within a single security domain:

- They may be used as part of the initial authentication of a principal (an initiator) for controlling access to the domain as a whole (the target). An example is authorisation based upon location of logon terminal or time of day.
- They may be used for controlling access to an operation, or group of operations, supported by the domain. An example is control of access of a principal (initiator) to an application or function of an application (target), or control of access by one application (initiator) to another application (target).
- They may be used for controlling access to an element, or group of elements, of the domain. Examples are access by a process (initiator) to a file (target), and access by a client process (initiator) to a field or record within a database (target).
- A combination of the above is possible.

When a system function is provided by a combination of security domains within a system, in particular a combination of service domains and platform domains, the overall system access control policy is enforced by the combination of authentication and authorisation services within all the constituent security domains.

This section describes the ISO model of access control described in detail in ISO/IEC 10181-3. This is based on the concept of making an access authorisation decision (the authorisation service) and then enforcing that decision.

5.4.1 Model

The basic principals and functions involved in access control are the initiator, *Access Control Decision Function* (ADF), the *Access Control Enforcement Function* (AEF) and the target. Figure 5-13 on page 82 illustrates the basic model of authorisation

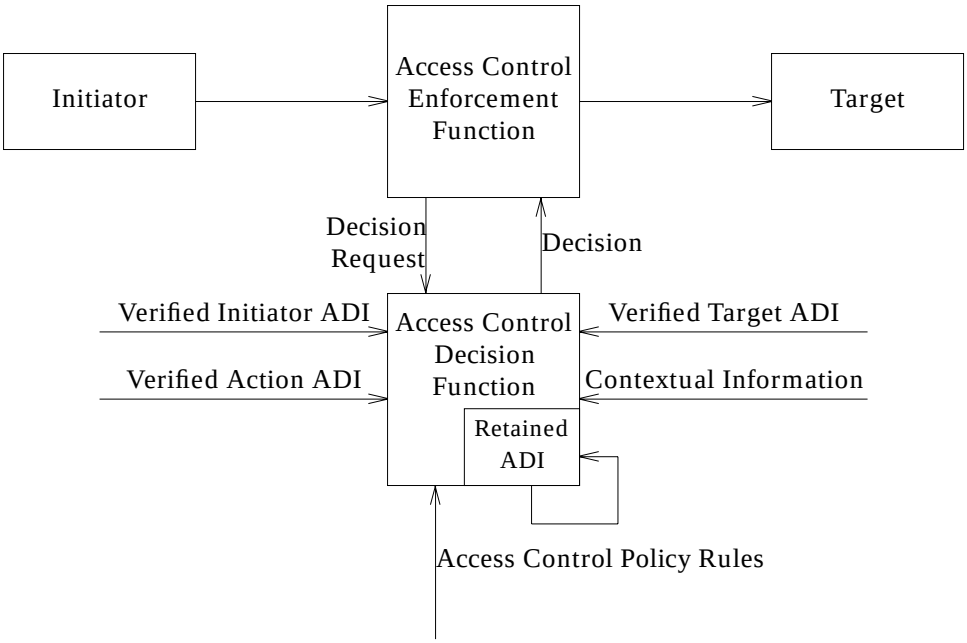

Figure 5-13 Basic Authorisation Service Model

Access Control Information (ACI) is information used for access control purposes. ACI may be associated with principals such as initiators or targets, may be associated with actions, and may include contextual information.

Access Control Decision Information (ADI) is the portion (possibly all) of the ACI associated with a principal or action that is made available for use in making a particular access control decision.

The AEF is responsible for ensuring that any actions by the initiator on the target are authorised (by the ADF). When an initiator makes a request to perform an action on the target, the AEF invokes the services of the ADF so that a decision can be made. To make the decision, the ADF is provided with, or acquires ADI associated with the initiator, the target and the action. The caller of the ADF is responsible for ensuring that the ADI has been verified before it is used as input to the access decision. Other inputs to the ADF are the access control policy rules and contextual information needed to interpret the ADI or the policy. Contextual information may include location, time of access or communication path.

When the AEF requests a decision of the ADF, the ADF requires access to the appropriate ADI and contextual information:

- ADI and contextual information may be preplaced at one or more ADF components after allocation of the ACI values.
- ADI may be delivered to the ADF as part of the decision request.
- ADI may be obtained from another functional element (for example, a Directory Service Agent). The ADI may be obtained by the initiator or target, or by the ADF itself.

As described in Chapter 3, the enterprise security authority as part of the system design procedure maps enterprise security policy rules to authorisation services supported by the system and maps enterprise security attributes to ACI within the system.

Examples of ACI are:

Initiator ACI

- individual access control identities
- identifier of hierarchical group in which membership is asserted, for example, organisational position
- identifier of functional group in which membership is asserted, for example, membership of a project or task group
- role that may be taken
- sensitivity markings to which access is allowed
- integrity markings to which access is allowed
- a target access control identity and the actions allowed on the target — that is a *capability*.
- security attributes of delegates
- location, for example, sign-on workstation.

Target ACI

- target access control identities
- individual initiator access control identities and the actions on the target allowed or denied them
- hierarchical group membership access control identities and the actions on the target allowed or denied them
- functional group membership access control identities and the actions on the target allowed or denied them
- role access control identities and the actions on the target allowed or denied them
- authorities and the actions authorised for them
- sensitivity markings
- integrity markings.

Action ADI

- ADI associated with operands of the action (data ADI), for example:
  - — sensitivity markings
  - — integrity markings
  - — originator identity
  - — owner identity
- ADI associated with the action as a whole, for example:
  - — initiator ADI
  - — permitted initiator and target pairs
  - — permitted targets

— permitted initiators

— allowed class of operations (for example, read, write)

— required integrity level.

Contextual Information

- time periods
- route (an access may be granted only if the route being used has specific characteristics)
- location (an access may be granted only to initiators at specific end-systems, workstations or terminals, or only to initiators at a specific physical location).
- system status (an access may be granted only for particular ADI when the system has a particular status, for example during a disaster recovery)
- strength of authentication, see **authentication method** on page 189 (an access may only be granted when authentication mechanisms of at least a given strength are used)
- other access currently active for this or other initiators.

ACI must be allocated to elements by the Security Authority. This is part of the process of installing or creating elements within the domain. ACI must be allocated to initiators before they may request any actions. ACI must be allocated to a target before it may be used. The allocation of ACI could require specific administrative action on the part of the Security Authority.

ACI could be configured so that a principal can utilise the services of a domain (for example, permitting a user to log in to a protected network), or the ACI could be allocated automatically as part of the element creation process, based on the security context of the action creating the element (for example, in a UNIX system, file creation permissions depend on *umask*).

ACI and ADI must be bound to the elements of a domain, including a sponsor and principal pair as an initiator, to provide assurance to the access control functions that the ACI or ADI is correctly associated with the element. The ACI, and its binding to an element, may be propagated. Therefore, the integrity of the ACI and its binding to an element requires protection when the ACI or the element is stored, processed or exchanged. The methods by which ACI may be bound to elements include:

- on the basis of storage location, for example, storage of security attributes as part of the inode of a file
- on the basis of principal identity combined with a security attribute security service; this may require the support of an authentication service to verify a principal's identity (see Section 6.2 on page 105)
- on the basis of cryptographic processes for the signing or sealing of sets of information; this is particularly relevant in communication services (see Section 6.3 on page 110).

It must be possible for the initiator to select a subset of ACI; the same should be true for a delegate.

If the ACI, or part of the ACI, associated with a principal is revoked, any ADI generated on the basis of that ACI must also be revoked. This can be done by specific action or by relying on the time expiry of such ADI when it takes the form of a security certificate or security token.

5.4.2 Classification of Authorisation Schemes

Both ECMA-138 and ISO/IEC 10181-3 present a classification of authorisation schemes which demonstrate that the various schemes can be considered as different parts of a continuous spectrum. This illustrates that the concepts applied are consistent. The variations occur where the ACI are held and managed. The basic schemes may be classed as:

Capability Scheme

ACI comprising target-name and access-type tuples assigned to initiators provides the basis of a capability-based authorisation scheme. Such a scheme is difficult to administer with a dynamic target population.

A generalised capability scheme may assign tuples comprising a target-group-name and access-type. This can provide authorisation to access types of target rather than specific targets. An example is the assignment to initiators of authorisations that grant authority to utilise certain functions with all appropriate targets. This use of capabilities within an operating system service context is often referred to as privilege. However, within this framework privilege has the wider meaning of access rights in a total sense.

Label Scheme

ACI associated with both initiators and targets together with an authorisation rule based on comparison of the initiator and target ACI provides the basis of a label-based scheme (sometimes referred to as a Mandatory Access Control scheme.)

Access Control List Scheme

ACI comprising initiator-name and access-type tuples assigned to targets provide the basis of an authorisation scheme based on access control lists (ACLs). Such a scheme is difficult to administer with a dynamic population of initiators.

A limited form of such a scheme is the permission bit scheme supported by the **XSH**, **Issue 4**, **Version 2** specification. The spectrum of authorisation schemes is illustrated in Figure 5-14 on page 86.

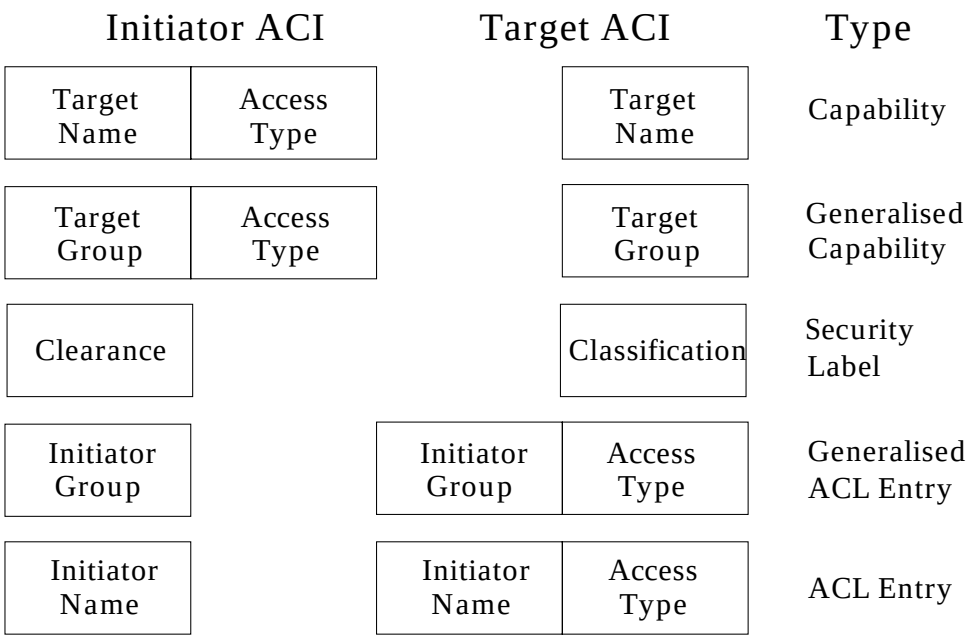

Figure 5-14 Spectrum of Authorisation Schemes

This demonstrates a gradual transformation from one concept into another. In particular the administrative effort and complexity is focused on initiators at the top of the figure and moves progressively to the targets at each layer. At the centre of the figure the administrative effort can be considered equally distributed between initiators and targets. At the bottom of the figure the administrative effort is focused on the targets.

In practice a combination of authorisation schemes is applied with the authorisation decision depending on the outcome of the combination of the authorisation decisions under each applicable scheme. This is illustrated in Figure 5-15.

Authorisation Scheme	Initiator ACI	Target ACI	Authorisation Scheme Decision		Final Authorisation Decision
ACL	Initiator Name →	ACL →	Yes / No →	Combination Algorithm	→ Yes / No
Label	Clearance →	Classification →	Yes / No →		
Capability	Capability →	Target Name →	Yes / No →		

Figure 5-15 Combination of Authorisation Schemes

The combination algorithm depends on the implementation. As an example, an algorithm may require access to be authorised by both a label-based scheme and an ACL-based scheme or by a capability-based scheme.

Figure 5-15 is an illustration of a combination algorithm.

Combination Algorithm									
ACL	N	N	N	Y	Y	Y	-	-	-
ACL-Capability	N	N	N	-	-	-	Y	Y	Y
Label	N	Y	-	N	Y	-	N	Y	-
Label-Capability	N	-	Y	N	-	Y	N	-	Y
Authorisation granted (G) or denied (D)	D	D	D	D	G	G	D	G	G

```
If((ACL.OR.ACL-Capability).AND.(Label.OR.Label-Capability))
    Authorisation granted
Else
    Authorisation denied
```

Figure 5-16 Example Combination Algorithm

In this example, authorisation needs to be granted by both the ACL authorisation scheme and the label-based scheme. Authorisation under the ACL scheme is granted by an appropriate entry in the ACL or by the initiator possessing an ACL-Capability. Authorisation under the label scheme is granted by the comparison of the initiator and target labels and the label scheme rule or by the initiator possessing a Label-Capability.

5.4.3 Discretionary and Non-discretionary Authorisation Schemes

Within a security domain an authorisation scheme may be a discretionary authorisation scheme or a non-discretionary authorisation scheme.

A discretionary authorisation scheme is one under which any principal using the domain services may be authorised to assign or modify ACI such that he may modify the authorisations of other principals under the scheme. A typical example is an ACL scheme which is often referred to as Discretionary Access Control (DAC).

A non-discretionary scheme is one under which only the recognised security authority of the security domain may assign or modify ACI for the authorisation scheme. This means that the security services within the security domain are responsible for the assignment of ACI when principals and data are created. A typical example is a label-based authorisation scheme used as part of an information flow policy based upon information sensitivity. Such a scheme is often referred to as Mandatory Access Control (MAC).

An authorisation scheme may have both discretionary and non-discretionary aspects. Thus a principal may be authorised to assign his own authorisations or a subset thereof to other principals or to make the set of authorisations more restrictive. An example under a label-based scheme is that principals may be authorised to increase the sensitivity of information, and thus potentially reduce the set of principals then authorised to access the information, but not to decrease the sensitivity of the information, and thus increase the set of principals then authorised to access the information.

5.4.4 Unverified ADI

An authorisation policy may permit the use of unverified ADI, perhaps representing an unauthenticated principal. A common example is anonymous ftp services in which both authenticated and unauthenticated principals are permitted to utilise the services of a system. The specification of a principal identity of anonymous permits access without requiring authentication and provides a default set of privileges.

A further example is the specific support for unauthenticated principals in the DCE implementation of ACLs.

5.4.5 Management Services

Within a system there may be several sets of the services described below for each of the distinct security domains comprising the system. As an example there is a set of such services for the platform domain, a set of services for particular applications, for example a RDBMS, and a set of services for support of a common access control service shared by distributed applications, for example, DCE.

Install-ACI

Install-ACI establishes an initial set of ACI (for example, capabilities for use by initiators, security labels for use by initiators and targets and ACLs for targets) associated with an element. This service may be integral with the creation of an element based upon the security context of the creating operation.

Modify-ACI

Modify-ACI modifies (for example adds, deletes or changes) the ACI associated with an element.

Revoke-ACI

Revoke-ACI revokes use of ACI relating to an element. This differs from Modify-ACI in that any ADI relating to this ACI is revoked.

List-ACI

List-ACI lists the ACI of a given element.

Disable-Component

Disable component disables the use of an access control function component. In the case of an AEF the service inhibits all access through the AEF.

Enable-Component

Enable component enables use of an access control function component.

Map-Text-String-to-ADI and Map-ADI-to-Text-String

These services translate the value of a named element of ADI between a text string representation and an internal representation and vice versa. These services are used by management applications and audit services.

5.4.6 Operational Services

The operational authorisation services operate within the context of a set of initiator ACI (privilege attributes). The acquisition and control of these privilege attributes are provided by the Domain Interaction Service as described in Section 6.2 on page 105, Section 6.3 on page 110 and Section 6.5 on page 125.

Operational services specific to the authorisation service are:

Get-Contextual-Information

This service gets the contextual information required for an access control decision to be made. This service is a supporting service for the ADF and is application and ADF specific.

Decide-Access

This service is provided by an ADF and determines if an access is allowed based on ADI presented to or acquired by the ADF. ADFs are generally particular to specific applications and sets of data, and may not be generally available to applications, for example, ADF functions within a database. Examples of more generic ADF services are *access*( ) supported by the **XSH**, **Issue 4**, **Version 2** specification to check the access of a process to a file, and the services provided to a DCE application by a DCE ACL Manager.

5.4.7 Impact on Primary Service APIs

A caller of a primary service API that is unaware of authorisation is illustrated in Figure 5-17.

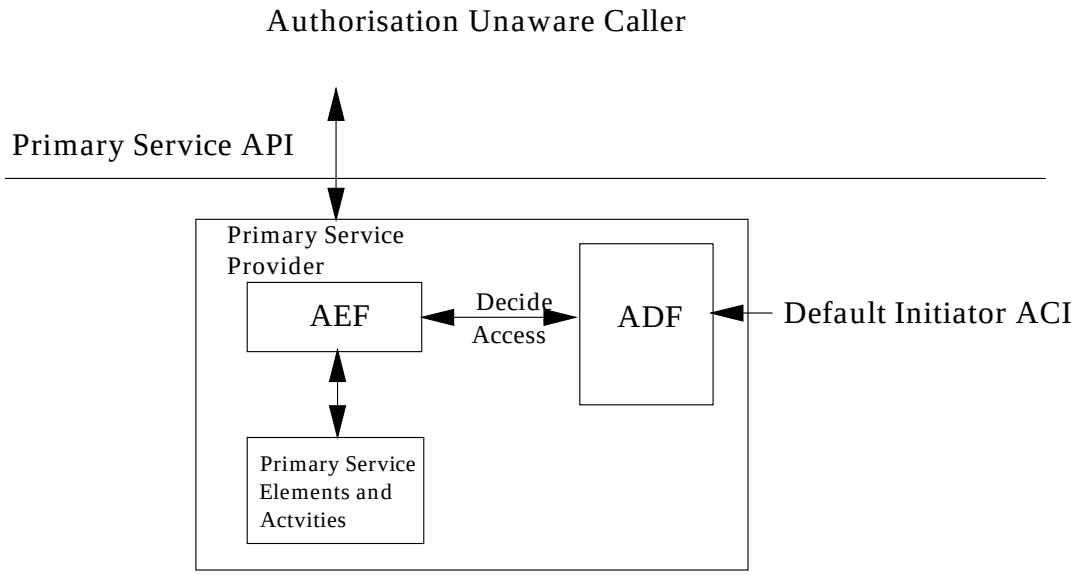

Figure 5-17 Authorisation Unaware Caller

In this case the primary service APIs do not allow the caller of the interface to assume responsibility for authorisation policy when invoking the API. The authorisation policy is enforced entirely by the primary service provider. The ADI and contextual information required by the ADF as the basis for the authorisation decision is acquired by the primary service provider (or the ADF). The default Initiator ADI bound to the tree of execution (of which invocation of the API forms a part) is used as input to the ADF. The primary service provider must implement the AEF but need not implement the ADF. An authorisation-selecting caller is able to select the authorisation security context of the service; the service provider enforces the authorisation decision. This is illustrated in Figure 5-18 on page 91.

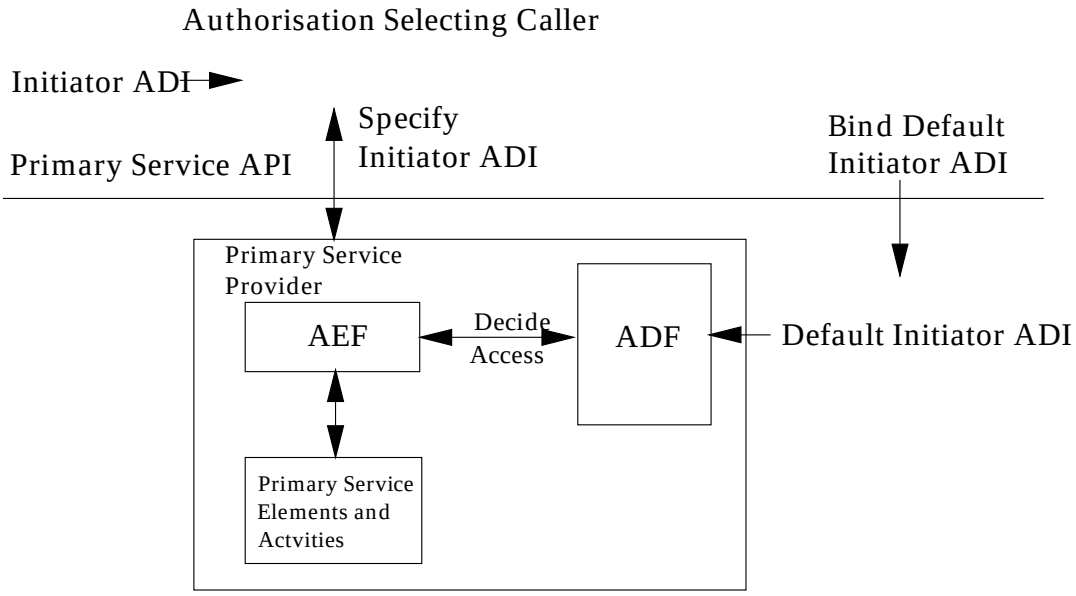

Figure 5-18 Authorisation Selecting Caller

In this case the caller of the API is responsible for selecting the Initiator ADI used in the authorisation decision. This initiator ADI can be passed directly to the primary service provider as parameters to the primary service API. Alternatively, it can be provided by means of a separate security service API to bind a particular set of initiator ADI with subsequent primary service invocations as a default. The primary service caller is responsible for managing and ensuring the security of the initiator ADI and its binding to a particular chain of service invocations.

An authorisation enforcing caller is responsible for enforcing authorisation as shown in Figure 5-19 on page 92.

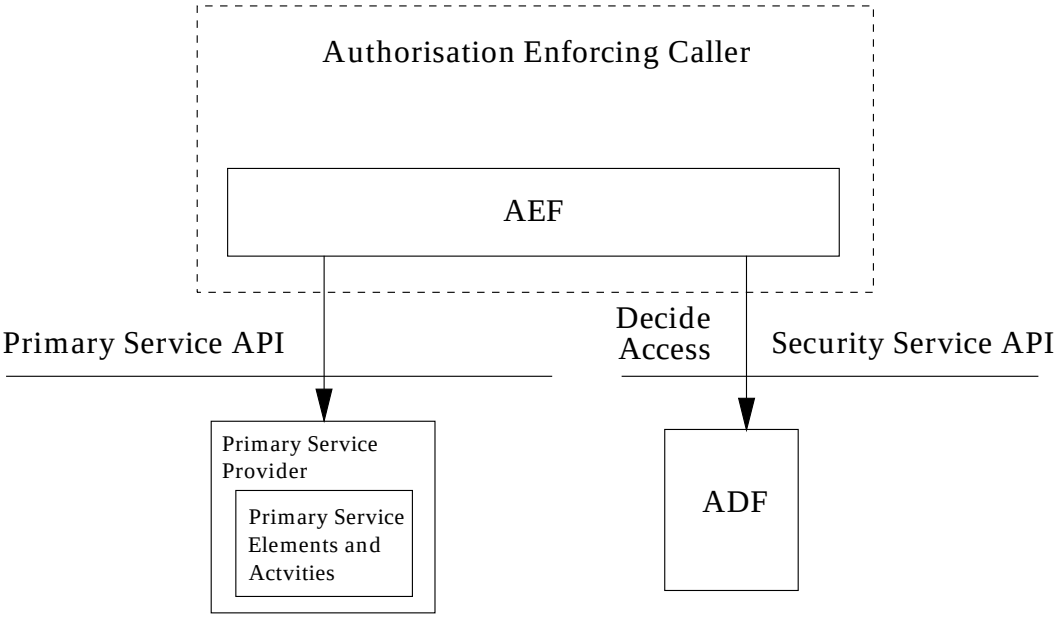

Figure 5-19 Authorisation Enforcing Caller

In this case the primary service provider does not enforce an authorisation policy on the use of its services. The caller of the primary service API is responsible for enforcement of authorisation policy and therefore contains the AEF. The caller can also contain the ADF; alternatively, it can call a separate service that provides the ADF capability.

In practice the primary service provider may be required to enforce an authorisation policy but on different data from the caller of the primary service. This is to ensure that only callers that are trusted with the responsibility to enforce the appropriate authorisation policy can invoke the primary service APIs. An example would be a platform service interface that requires the calling process to possess an appropriate capability.

Note: This model can also apply if the caller is permitted to assume responsibility for authorisation policy normally enforced by the primary service provider. In this case the Primary Service Provider surrenders authorisation decisions to the caller.

5.4.8 Relationship to Other Security Services

Figure 5-20 provides an example of how the authorisation services interact with the other security services. This is based on a *pull model* in which the server retrieves the privileges of the initiator from a privilege attribute service based on an authenticated identity.

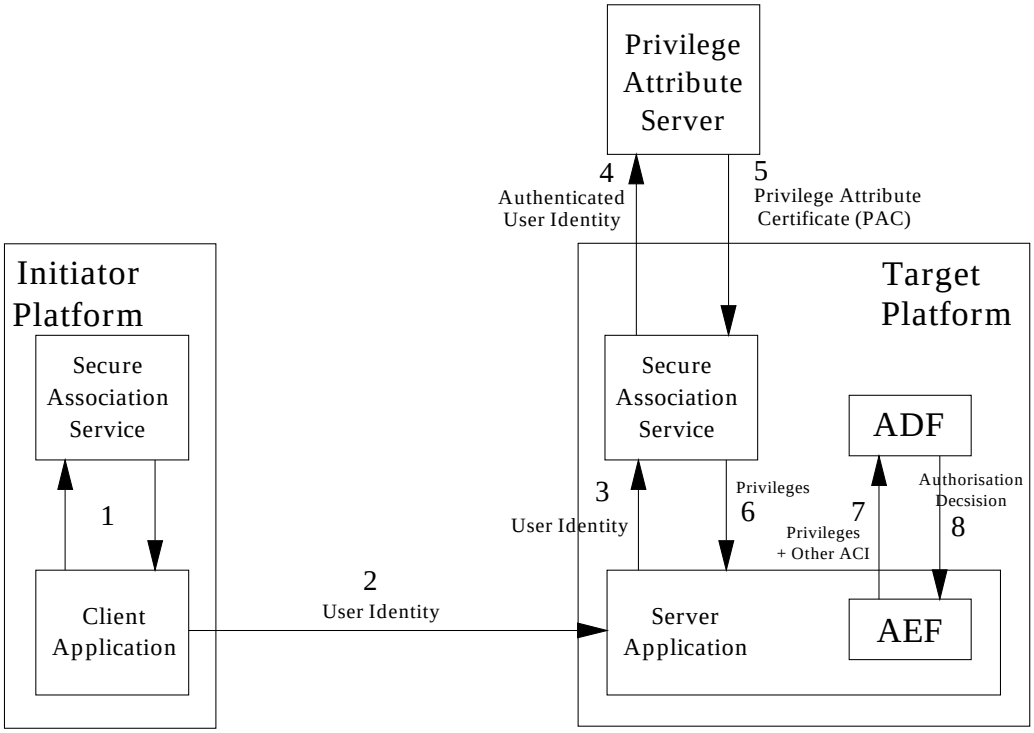

Figure 5-20 Authorisation and Other Security Services

The client application (initiator) securely transmits the identity of its principal (a user) to the server application by means of the secure association service (steps 1, 2 and 3 in the diagram.). Refer to Section 6.3 on page 110.

The secure association service retrieves the principal's privileges from the privilege attribute service (steps 4 & 5) and makes these available to the server application (target) (step 6.) Refer to Section 6.5 on page 125 and Section 6.4 on page 123.

The server application invokes the ADF passing the principal's privileges and other necessary ACI, such as the target ADI (for example ACL) and contextual information, to the ADF (step 7.)

The ADF returns the authorisation decision to the sever application (step 8) which is then responsible for enforcing the decision.

This is only one example of the possible deployment of the different security services. A similar example is a *push model* where the initiator retrieves its own privilege attributes and uses the secure association service to propagate them securely to the target.

5.5 Security Audit

A security audit service detects security-relevant actions within an IT system. When such an event occurs, the audit service generates an audit event, which can be recorded, reported, archived and analysed. Configurable selection criteria determine which events are recorded and reported within a security domain. These criteria are defined by the security policy applicable to the domain.

Security audit services place several dependencies on authentication, authorisation and access control measures.

This service description does not encompass a non-repudiation service except to the extent that the recording of security audit information supports such a service.

5.5.1 Model

Figure 5-21 on page 95 illustrates the security audit services model which comprises the following components.

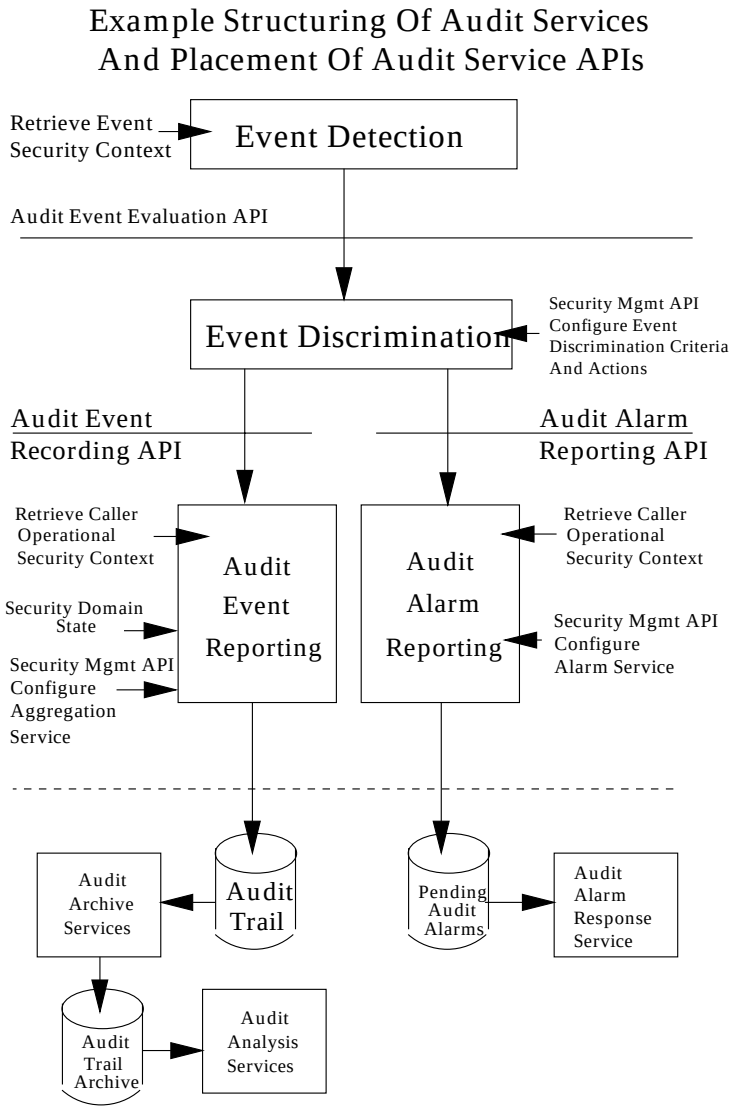

Figure 5-21 Basic Security Audit Service Model

Event Detection

This component detects each security-relevant operation, and generates an audit event containing a description of the operation and information from the local security context. Copies of this component are incorporated as needed within primary service providers and security audit detecting callers (see Section 5.5.10 on page 101).

Event Discrimination

This component receives all audit events and determines their disposition based on configurable criteria and reactions. Depending on which criteria an audit event matches, it may or need not be recorded, and may cause an alarm.

Event Recording

This component receives audit events for recording. The recording service may add information from the caller's operational security context and the security domain state to the audit event. The event is added to the appropriate audit trail at a configurable destination.

Alarm Reporting

This component receives audit events configured as alarms. They must either immediately initiate an alarm or be added to a collection of such events subsequently to trigger an alarm when a configured threshold of events has occurred. The alarm reporting service may add information from the caller's operational security context as it generates an alarm from the audit event. When triggered, an alarm is sent to a configurable destination.

The triggering of an alarm may also initiate an action to protect the system from further threat, for example, by inhibiting the operation causing the generation of the audit events triggering the alarm. An example could be the invocation of an Invalidate AI service or Invalidate ACI service, or both to prevent further use of a principal identity or I/O device in the case of failed or duplicated authentication operations.

Alarm Examination

Audit alarms generated must be examined and responded to by a representative of the security authority of the security domain, usually the security auditor.

Audit Archiving

This component allows audit trails to be saved for later analysis.

Audit Analysis

This component retrieves and merges archived audit trails and produces reports.

5.5.2 Distributed Audit Services

Within this framework an IT system comprises many security domains. An audit service is generally supported as one or more specialist security domains providing all or parts of the total audit service to all, or a subset of, the other security domains.

The specific operations requiring auditing vary according to the function of the security domain. Additionally if operations span security domains under the ultimate control of different security authorities, some mapping of security attribute syntax and semantics may be required.

To support the centralised archiving and analysis of audit trails in a distributed heterogeneous environment it is necessary to establish a common audit trail interchange format, including a common interpretation of audit event types.

A common interpretation of audit event types is best achieved by considering security-relevant actions at the abstract level. Examples of such abstract actions are:

- association creation and termination

- security-relevant data creation and deletion
- security-management actions (for example install, deinstall, disable and enable).

An audit trail export function is then required to convert any local representation of an audit trail to a common interchange and analysis format.

Note: A centralised audit analysis service can provide an analysis at a suitably abstract level. If security exceptions are identified, a lower level of audit analysis can be undertaken. Eventually it may be necessary to revert to source machine format for a detailed analysis.

An interchange format could comprise a common interchange header and detail record together with an opaque source format record.

5.5.3 Management Services

Management services fall into three groups: configure audit event discrimination, configure audit alarm reporting, audit archiving and audit analysis as described in the following sections.

5.5.4 Configure Audit Event Discrimination

Install-Audit-Event-Discrimination-Criteria-and-Reactions

Installs the criteria used to control which audit events are generated and the consequential action. For example, criteria may include:

- principal requesting the operations
- sensitivity of the operations
- attributes of the information being processed
- context of the operation, for example, location and time of day (to detect unusual behaviour).

Examples of consequential action include no action, generate alarm, record event, or record event and generate alarm.

Modify-Audit-Event-Discrimination-Criteria-and-Reactions

Modifies the criteria and actions used in event discrimination.

Deinstall-Audit-Event-Discrimination-Criteria-and-Actions

Removes selected criteria and actions so that they no longer affect event discrimination.

5.5.5 Configure Audit Alarm Reporting

Install-Audit-Alarm

Installs instructions that determine what happens when alarms are received. This requires the specification of such information as threshold values, destinations of alarm message, contents of alarm message, and actions to be taken in addition to sending alarm (for example, disable user account and terminal).

Modify-Audit-Alarm

Modifies what happens when alarms are received.

Deinstall-Audit-Alarm

Removes instructions for selected alarms.

Disable- and Enable-Audit-Alarm

Temporarily turns off or on alarm reporting for a selected alarm.

5.5.6 Configure Audit Event Recording

Install-Audit-Trail

Initialises an audit trail and designates it as the destination for recorded audit events.

Reinstall-Audit-Trail

Stops sending events to the audit trail.

Disable- and Enable-Audit-Trail

Temporarily stops or restarts adding events to the selected audit trail.

5.5.7 Audit Archiving

Archive-Audit-Trail

Saves an audit trail for long term storage. The audit trails must be securely archived to allow for subsequent analysis, possibly after a long elapsed time.

Export-Audit-Trail

Converts an audit trail to standard interchange format and copies it to a given destination.

5.5.8 Audit Analysis

Import-Audit-Trail-Archive

Retrieves for analysis a copy of an audit trail previously archived from a given source.

Merge-Audit-Trail

Allows the merging of several audit trails from given sources and the tracing of chains of events.

Produce-Reports

Provides measures for analysing the audit trails produced and generating useful reports on security-relevant usage and operation. Procedures, mechanisms and tools are required to ensure that records of security-sensitive events are regularly analysed. Similarly, procedures and mechanisms are required to ensure that business-sensitive events are subsequently available to support the resolution of legal and contractual disputes.

5.5.9 Operational Services

Audit-Event-Detection

The detection of events must occur within the software providing the services that constitute the security-relevant operation. That is, event detection is part of the primary service provision and must be included within the specification of that service provision. This means that an API specification for the primary service shall include any requirement for the generation of an audit event.

The audit events generated must include an assertion of the identity of the presenter of the event.

Audit event detection must be included as a fundamental part of a primary service provision. However, the security policy can permit the security authority (as represented by the Security Auditor) to configure the actions to be taken on the detection of an audit event based on local or current context. A received event therefore has to be screened against the actions configured for the event.

Audit-Event-Discrimination

Audit-Event-Discrimination must verify that the events it receives are from valid sources. This prevents attacks that attempt denial of service by overloading the security audit services.

Audit-Event-Recording

An event that requires recording must be appended to an audit trail in a manner such that it cannot be subsequently modified or deleted. For each event the identity of the initiating principal or performing principal or both, together with other relevant information, such as date and time, are recorded.

Other information that must be recorded includes the operational security context of the event together with security domain management information that permits the record to be interpreted (for example, mappings of internal security attribute values to text strings). This may include the recording of cryptographic keys used for encrypting or sealing data. For efficiency, the security domain management information may be recorded periodically (for example, once per day) and the security management state applicable to any individual event regenerated by tracing all audited changes to that information since the last record of the state.

Audit-Alarm-Reporting

An event that is configured to generate an alarm must either immediately initiate an alarm or be added to a collection of similar events subsequently to trigger an alarm when a configured threshold is reached. The speed with which the alarm is raised is determined by the severity of the event. Ideally, the reporting of the alarm should give as much time as possible to those responsible for limiting the potential damage. The alarms are reported over one or more robust communication services to designated user interfaces.

Audit-Alarm-Response

Audit alarms generated must be examined and responded to by a representative of the security authority of the security domain, usually the Security Auditor.

Suspend-Primary-Service-Auditing

See Figure 5-23 on page 102.

5.5.10 Impact on Primary Service APIs

If a primary service provider supports services that can be interpreted as security auditable events by the security policy, the primary service provider must incorporate the audit event detection capability and invoke the other audit services when an auditable event is detected. The specification of the primary service APIs should include the requirement to audit the API. The caller of the security action may be unaware of any auditing requirements. The primary service implementation, or the audit services it invokes, must derive all necessary security context information for inclusion in the audit record, as shown in Figure 5-22.

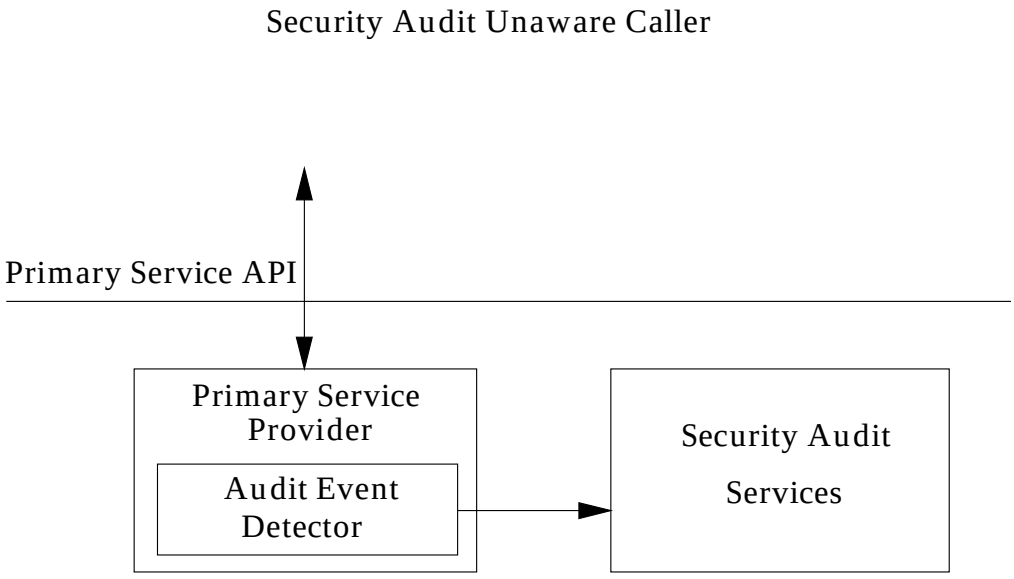

Figure 5-22 Security Audit Unaware Caller

Figure 5-23 on page 102 illustrates a caller enforcing security audit.

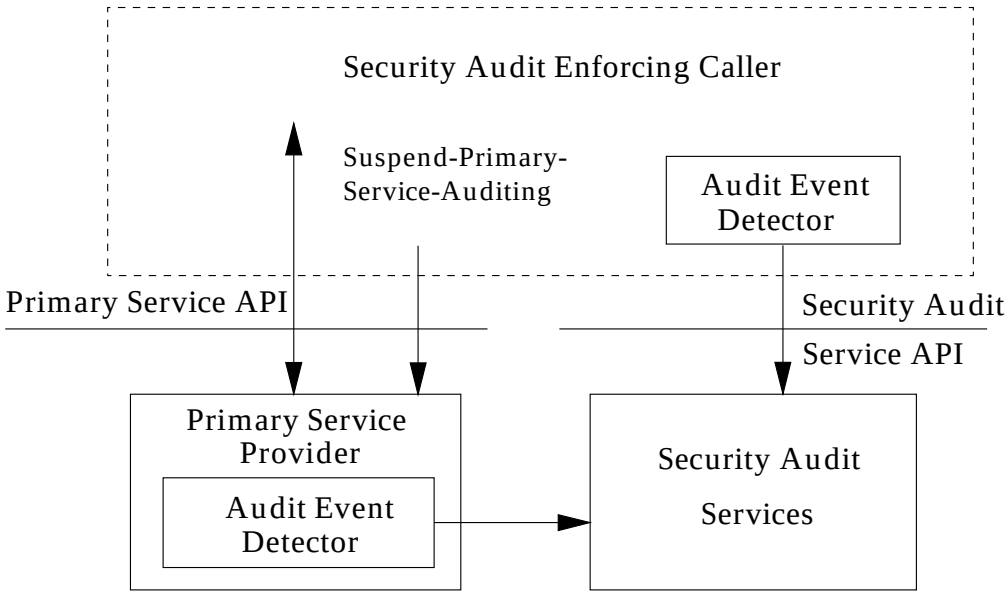

Figure 5-23 Security Audit Enforcing Caller

If the caller of a primary service API also provides services that are interpreted as auditable events, it must incorporate audit event detection and invoke the other audit services. In this case the security policy may permit the caller of the primary service API to modify the event discrimination configuration for the primary service to suppress the generation of any audit records or alarms from its use of the primary service.

Chapter 6

Domain Interaction Security Services

Chapter 4 introduces the concept of classes of security services and Chapter 5 describes the basic security facilities. These basic security facilities must operate within the context of a set of security information.

This chapter describes the Domain Interaction Security Services which manage and propagate that security information context; they are illustrated in Figure 6-1.

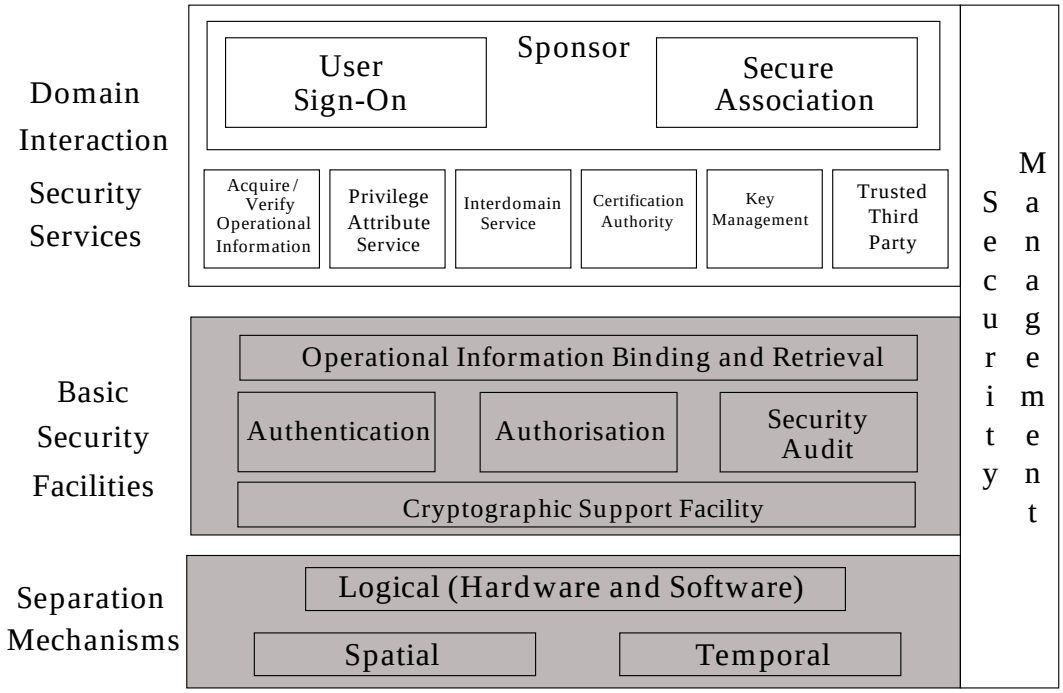

Figure 6-1 Domain Interaction Security Services

These services comprise:

- User sign-on service
- Secure association service
- Acquire/verify operational information service
- Privilege attribute service
- Interdomain service
- Certification authority service
- Key Management service
- Trusted third party services.

6.1 Sponsor

A Sponsor service promotes a principal within a security domain. Sponsor services are deployed within each security domain. The sponsor service is responsible for managing the formation of associations between its own security domain and other security domains.

Sponsor services are responsible for:

- the initial identification and authentication of the principal. For this purpose the sponsor uses the appropriate authentication service.
- establishing one or more security contexts for the principal

  The security contexts contain security attributes to be used in subsequent security operations; typically these attributes are: principal identification, audit identification and access control information. The Sponsor services use the Privilege Attribute Service to acquire the relevant security information based on the principal's authenticated identity.

  This could include establishing a platform domain security context represented by a set of process attributes if platform services are used to enforce policy.
- facilitating interactions between the principal and the other services within the security domain

  In certain cases the sponsor may be assigned responsibility for mediating access to these services; in this respect it operates as an Access Control Enforcement Function.

A platform domain that supports services for individuals has a sponsor service provided by a dedicated service domain. This service domain is referred to as a User Sponsor service and is responsible for the formation of associations between users as principals and the services of the platform hosting the primary interface with that user; it is responsible for managing the user session. The User Sponsor services are outlined and described in more detail in Section 6.2 on page 105.

The server component of a distributed application also includes sponsor services that are responsible for controlling the formation of associations by client components with the server component of the application. This requires:

- the invocation of the secure association service (see Section 6.3 on page 110) to support the secure propagation of the requested security context for the association
- the authorisation of the association creation on the basis of the requested security context
- the binding of the client principal's privilege attributes to the services of the server service domain.

6.2 User Sign-on

Users require access to an IT system but are physically distinct from it. To utilise the services of an IT system a user must create an association between himself as a principal and the IT system. The creation of such an association is the User Sign-on procedure.

The sign-on model presented here identifies a set of operations that a comprehensive distributed sign-on system would support. This sign-on model may be profiled by specific implementations. A sign-on service may be minimally a purely local operation (for example, requiring a user to provide a password to enable a Privacy Enhanced Mail user agent to access a user's private key.

The user interfaces to the system by means of a User Sponsor service. The User Sponsor service is responsible for promoting the user to the appropriate authentication and privilege attribute services and establishing the environment of the user session, which then provides the user with services. This procedure is illustrated in Figure 6-2.

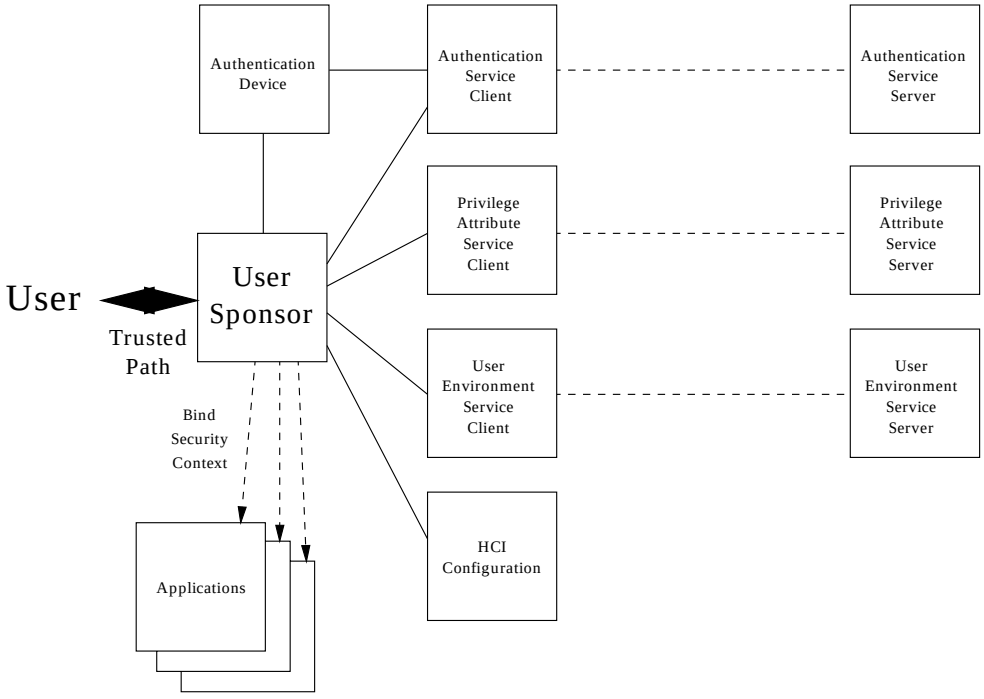

Figure 6-2 User Sign-on Model

Trusted Path

Platform domains, through which users interact with an IT system, must direct all initial user contact with the system to a user sponsor service domain. The system security policy can demand that this interconnection is by a trusted path.

The purpose of the trusted path is to provide assurance of the integrity and confidentiality of the communication between the user and trusted components of the system. An authentication trusted path is a mechanism supported by the platform domain that provides the user with assurance that communication is only with the user sponsor. It also provides the user sponsor with assurance that it is communicating directly with a user I/O device.

User Authentication

The user sponsor service is responsible for promoting the user to the authentication service and, if successfully authenticated, for promoting the authenticated user to other services of the system. This procedure is detailed in Figure 5-9 on page 71. The user sponsor could support different authentication schemes and mechanisms and should be configured to use the correct scheme.

User Session Security Context

If the user authentication is successful the user sponsor is responsible for promoting the authenticated user to the privilege attribute service to obtain the set of initiator privilege attributes. These attributes are in the form of a Privilege Attribute Certificate (PAC), that represents the security context of the user session. This procedure is described in Section 6.5 on page 125. The user sponsor can make repeated calls on the privilege attribute service during a user session and obtain modified privilege attributes. For example, the user could select specific security attributes (ADI), such as a sensitivity label, role or group, from the total set of security attributes (ACI) available to the session. The modified PAC (ADI) is then used as the security context for specific applications that are invoked.

User Session Environment

The user sponsor is responsible for establishing the environment of the user session, for example, home directory, environment variables and list of applications accessible by user. These are provided by a user environment service.

Note: Although these are not considered a security service they are included here for completeness in the description of the sponsor services.

User Sponsor HCI

The interface between the user sponsor and the user could also be configurable, for example, the messages displayed, session time-out intervals, and so on. This information is retrieved from the user sponsor SMIB.

6.2.1 Management Services

Management Services fall into two groups: user sponsor service configuration and user environment service management services, as described in the following sections.

6.2.2 User Sponsor Service Configuration

The user sponsor management services include the following.

Configure-Authentication-Service

This configures the user sponsor to use a particular authentication service.

Configure-Privilege-Attribute-Service

This configures the user sponsor to use a particular privilege attribute service.

Configure-User-Environment-Service

This configures the user sponsor to use a particular user environment service.

Configure-User-Sponsor-Security-Policy

This configures the user interaction security policy aspects of the user sponsor services, for example, messages to be displayed, user interaction time-outs, and so on.

6.2.3 User Environment Service Management Services

The User Environment Service (UES) requires management services as follows.

Install-User-Environment

This installs an environment for a user in the UES service. A user environment may include a list of applications, a home directory, user session process environment variables and cultural information such as locale.

Deinstall-User-Environment

This removes a user environment from the UES.

Modify-User-Environment

This modifies a user environment previously installed in the UES.

List-User-Environment

This lists the components of a user environment or set of user environments.

Map-User-Environment-to-Text-String

This maps the value of a component of a user environment to a text string.

Map-Text-String-to-User-Environment

This maps a text string to the value of a user environment component.

6.2.4 Operational Services

The user sponsor operational services are HCI services including user authentication services, user application services and trusted path application services, as described in the following sections.

6.2.5 User Authentication Services

Solicit-User-Claim-AI

This requests the authentication information required from the user.

Change-Authentication-Credentials

This service provides a user interface to the Change-AI service of the authentication service to permit the user, for example, to change password.

6.2.6 User Security Context Services

These services provide an HCI for the user to manage security contexts. Some of these services could be incorporated, or hidden, within the user application services described below.

Create-Security-Context

This service allows a user to select specific privilege attributes from the set of security attributes within the security context of the user session used when executing an application. For example, a user can select a sensitivity label from the user session clearance, a role from the set of roles available, a group from the set of groups available. This service provides a user interface to the operation of selecting ADI from a set of ACI.

Display-Current-Security-Context

This service displays details of the current security context within which the user is operating. An example is the *whoami* command. This service could be supported by the continual display of security context information as a security label on the display. For example, a security-enhanced version of X Windows labels each client window with a security label.

List-Security-Contexts

This service lists the set of security contexts that the user previously created and that are available for use.

Select-Security-Context

This service sets the selected security context as the default security context for subsequent operations.

Delete-Security-Context

This service deletes a security context from the set of security contexts available to the user within the current session.

6.2.7 User Application Services

These services allow a user to manage applications. The user sponsor may have to constrain the applications that can be invoked by a user to only those configured for the user. In other words, the user sponsor may be responsible for enforcing access control on a user's access to applications.

List-Applications

This service lists the set of applications configured as available to the user.

Invoke-Application

This service invokes the selected application within the default security context. A security context could also be specified as a parameter to this service.

Suspend-Application

This service lets a user suspend an application so that he can interact with another.

Resume-Application

This service lets a user resume a previously suspended application.

Terminate-Application

This service lets a user terminate an application.

6.2.8 Trusted Path Application Services

These services let a user manage applications that must be executed from a trusted path. For example, an administrator may have to execute security administration applications from a trusted path.

Invoke-Trusted-Path

This lets a user invoke a trusted path for executing applications that require a trusted path.

List-Trusted-Path-Applications

This service lists the set of applications available to the user.

Invoke-Trusted-Path-Application

This service invokes the selected application within the default security context. A security context could also be specified as a parameter to this service.

6.3 Secure Association Service

An essential part of the operation of an IT system is the formation of associations. Section 6.2 on page 105 describes the secure formation of associations between Users as principals and the IT system. This section deals with the formation of secure associations between peer principals within the IT system; these principals are the sponsors within security domains.

The creation of a secure association is subject to the *secure interaction policy*. The associations permitted between specific security domains are configured by management action prior to their use using the secure association management services. The secure association operational services actually create the security context for an association between the security domains. The establishment of a secure association depends upon several other security services:

- Authentication and privilege attribute services provide a set of initiator operational security information in the form of initiator credentials.
- Acquire/verify operational information services generate operational security information in the form of a secure association context to be exchanged between the principals.
- Communication security services, if available, establish a communication channel for the association that meets the requirements of the secure interaction policy with respect to communication confidentiality and integrity protection.
- Authorisation services confirm that the requested association is permitted under the secure interaction policy.
- Interdomain services map the semantics and syntax of security attributes if the association is between services located in security domains with different security attribute semantics and syntax.

The creation of an association between one set of principals and data within an IT system requires the formation of an association between principals that contain the primary associating data. Thus an association between service domains also requires the formation of an association between platform domains (unless collocated within a single platform domain). This is illustrated in Figure 6-3 on page 111.

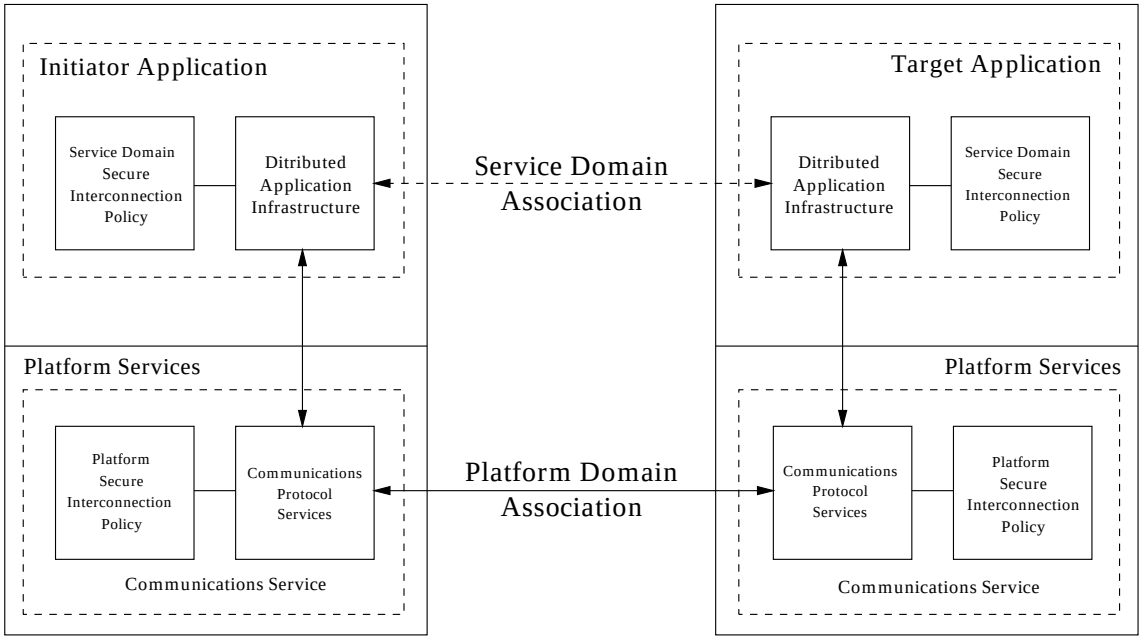

Figure 6-3 Nested Structure of Associations Between Security Domains

The creation of the association is subject to the secure interaction policies applicable to each of the constituent associations required. This equates to associations being formed and secure interaction policy requirements applying to different communication service layers (for example layer 7 and layer 4 of the OSI model).

6.3.1 Model

The stages in the creation of a secure association are illustrated in Figure 6-4 on page 112. Examples from the operation of DCE are given to illustrate each stage. A discussion of the secure association service in the context of a store and forward service is included in Section E.2 on page 178.

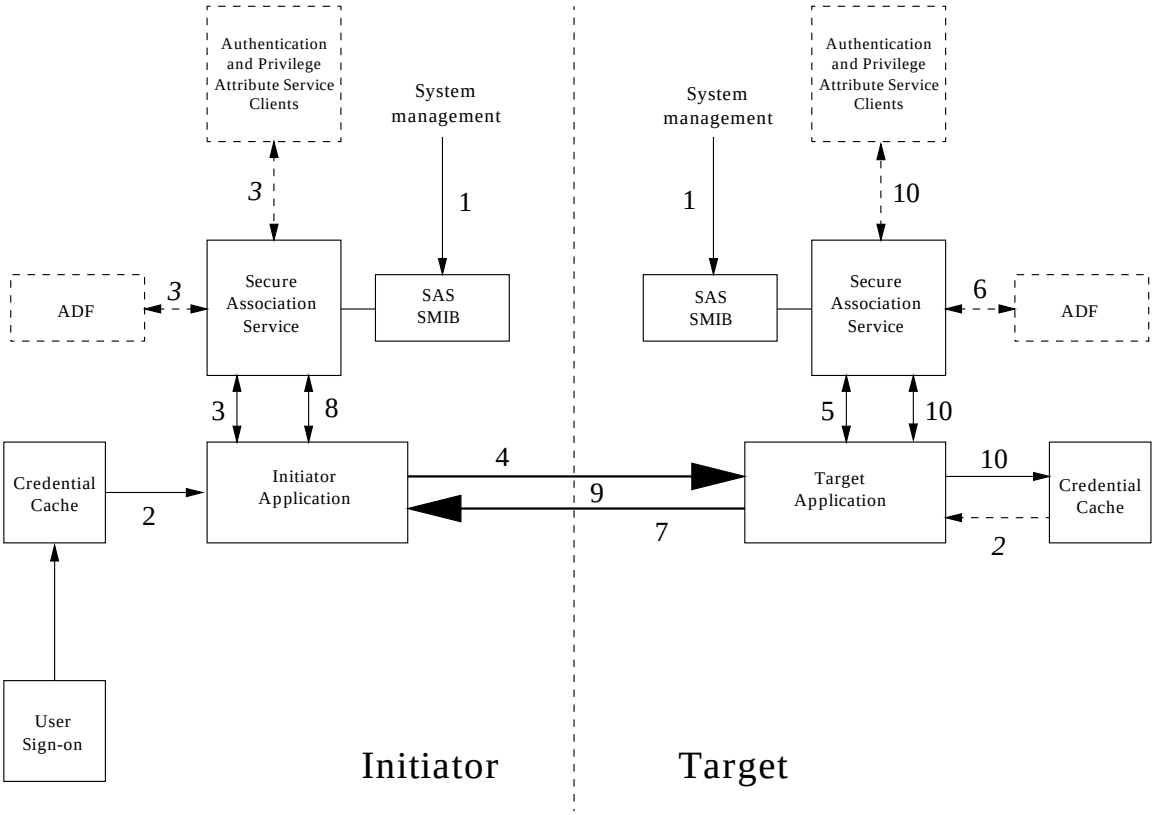

Figure 6-4 Secure Association Service Model

1. **Establish-Secure-Interaction-Policy**

   An association can only be created within the bounds of a secure interaction policy established for the formation of a particular association (or set of associations). The detailed rules of the secure interaction policy must be interpreted into a set of criteria for the SMIBs of the interacting security domains. This includes any requirements for an interdomain service to map the syntax and semantics of security attributes between their representations within each security domain, or to a common interchange format (see Section 6.6 on page 128.

   Example: The environment must be set up so that the DCE client and server can talk to each other, for example, configuring the target server in the initiator cells' registry and setting ACL entries to allow the initiator access to the target.

2. **Initiator-Credentials**

   A sponsor of a service domain initiating a secure association must possess credentials such as one or both of the following:

   - operational security information such as Exchange AI or initiator ACI appropriate to the principal receiving service from the service domain.
   - operational security information such as Exchange AI or initiator ACI appropriate to the service domain itself as a principal.

Operational security information appropriate to a user as principal is derived by the user sponsor service as part of the user authentication process. It is propagated securely within the user session service domain operating on behalf of that user principal. Security information appropriate to a service domain itself as a principal is established when the service domain is created by the platform domain with initial information (for example, service domain identity, cryptographic key or security certificate).

A service domain providing services on behalf of many principals may possess more than one set of initiator security information and may select which is to be used for the creation of a particular association.

The default credentials associated with the tree of execution of the initiating security domain can be used. Alternatively, the initiator can request to modify the credentials, for instance to support delegation of some of the initiator's access rights to the target.

Example: The Initiator Credential in DCE is the Privilege Ticket-Granting Ticket (PTGT). A client can invoke the DCE Privilege Server (Privilege Attribute Service) and request a PTGT for a service. The PTGT contains only a subset of the privileges (that is DCE groups) available to that client.

3. **Generate-Association-Security-Context**

   The initiator invokes the Generate-Association-Security-Context service to generate the operational security context of the association. The initiator can specify particular aspects of the security context; otherwise it can accept, or have imposed, specified defaults under the secure interaction policy. This includes specification of the Cryptographic Quality of Service (confidentiality and integrity services) and the strength of authentication mechanisms used.

   The operational security context for the association is returned to the initiating sponsor as a security token.

   Example: As an example, within DCE the client agent may obtain a valid ticket from its credential cache. If such a ticket does not exist the client uses its PTGT to request a ticket from the Ticket Granting Service (TGS) of the Key Distribution Centre (KDC).

   The KDC decodes and verifies the Client's request. To verify the request the KDC obtains it own secret key from the registry, reads the decoded request, decrypts and decodes the ticket and authenticator.

   The KDC unparses the server's name specified in the client's request, obtains the server's secret key from the registry, generates the random session key, encrypts the server ticket and reply, and sends them to the client.

4. **Initial-Security-Information-Exchange**

   If the generation of an association security context is successful, the initiator sends the security token representing the requested security context for the association to the sponsor within the target service domain.

   Example: Within DCE the Client Agent sends the request consisting of three components (the ticket, the authenticator and the request body) which are cryptographically linked. The request body is left empty.

5. **Target-Verify-Association-Security-Context**

   The target service domain sponsor invokes the Verify Association Security Context service, with the security token received from the initiator as a parameter. This service:

- verifies the authenticity of the security token by invoking the operational information verification service; this may in turn require the invocation of certification authority services if a certification chain has to be verified
- if necessary, identifies the interdomain service appropriate to the association, establishes the communication security services required by the association security context and returns an association security context to the target sponsor.

Example: Within DCE the Server decodes and verifies the Client's request. To verify the request the Server obtains its own secret key from the registry, reads the decoded request, decrypts and decodes the ticket and authenticator.

6. **Authorise-Association**

   The sponsor, having verified the authenticity of the association security context:

   - verifies the identity of the initiator
   - verifies the authorisation for the target to form an association with the initiator with the requested security context. This is based upon operational security information included within the association security context and security information possessed by the target. Verification of authorisation may include verification of the chain of delegation used to request this association.

   Example: Within DCE the Server accepts or rejects the association on the basis of the Ticket and the Authenticator.

7. **Responding-Security-Information-Exchange**

   If the association is authorised, the target sends the security token generated by the target verify security context operation to the initiator. This conveys security information regarding the acceptance or otherwise of the originally specified security context and levels of service to be used within the secure association.

   Example: Within DCE the application server's reply to the client is sent encrypted with the client-server key. An operation id is assigned.

8. **Initiator-Verify-Association-Security-Context**

   If required, the initiator sponsor invokes the generate security context service with the security token returned by the target service domain. This service performs the same functions as the verify association security context service in the target domain but for the association security context returned by the target domain sponsor.

   Example: This is not used within DCE.

9. **Association-Security-Context-Negotiation**

   In some circumstances it may be necessary for the initiator and target to negotiate over some aspects of the security context establishment. In this case further calls to the operational services of the secure association service and further exchanges of security tokens may occur.

   Example: This is not used within DCE.

10. **Extract-Security-Information-from-Association-Context**

    To enforce security policy within its own security domain the target sponsor extracts the operational security information, such as access control information and audit identity and audit discrimination information, from the association security context. The target sponsor may utilise the operational information bind service to bind that operational

security information with the tree of execution within the target domain. This may include mapping the operational security information to platform domain process attributes.

Example: The interface *rpc_binding_inq_auth_client*( ) enables a DCE server to retrieve a client's DCE PAC from the client's login context.

The above description is based on the assumption that the secure interaction policy requires an association to be initiated by an authenticated principal. However, a secure interaction policy may allow an association to be initiated by an unauthenticated principal (for example, as in anonymous ftp) with the target being required to authenticate itself to the initiator. The target may also need to specify the QOP to be used for the association with the unauthenticated initiator.

6.3.2 Management Services

These services manage the secure interaction policy applicable to potential associations. That is, these services do not create actual associations but manage the capability for associations to be formed.

Configure-Default-Association-Policy

This service configures those components of a secure interaction policy that are to be applied as defaults to all permitted associations.

Install-Association-Policy

This service installs a permitted initiator and target pair together with an associated permitted security context as defined by the secure interaction policy. This includes the specification of aspects such as the quality of protection (QOP) used within the association and the strength of authentication mechanisms employed. It may also include the configuration of an interdomain service to map the semantics (and possibly the syntax) of a set of security attributes within another security domain into the semantics (and syntax) of the security authority's own security domain.

Modify-Association-Policy

This service modifies the details of a permitted association.

Deinstall-Association-Policy

This service deletes a permitted association.

Disable-Association-Policy

This service sets a permitted association state such that it is unavailable for use.

Enable-Association-Policy

This service terminates the disabled state of a permitted association.

Revoke-Association

This service revokes an association. It can be invoked if the permitted association on which it is based is disabled. It can also be invoked if ACI on which authorisation for the association is based is itself revoked. This service is difficult to support in a store and forward service such as X.400.

6.3.3 Operational Services

Generate-Association-Security-Context

An initiator uses this service to request the generation of a set of operational security information representing the security context, for use in creating an association with a specified target. It can also be used to verify the authenticity of a set of operational security information returned by a target.

It requires the input, either specifically or as default, of the initiating principal's credentials and possibly a set of optional facilities requested by the client. For example, these optional facilities could include the delegation of some of the principal's access rights to the target service domain. The delegated access rights could be used in forming further associations to obtain service from other service domains. An example of this service is *GSS_Init_sec_context*( ) in the the **GSS-API Base** specification.

Verify-Association-Security-Context

A target uses this service to verify the authenticity of the set of operational security information provided by an initiator and that defines the requested security context. It generates a set of operational security information for return to the initiator including an indication of which options requested by an initiator are supported by the target. An example of this service is *GSS_Accept_sec_context*( ) in the the **GSS-API Base** specification.

Export-Security-Context

This service returns the security context referenced by the specified handle in a format suitable for export to another entity.

Import-Security-Context

This service imports a previously exported security context and returns a handle to refer to it.

Release-Association-Security-Context

This service releases and deletes a security context created by a generate or verify service on termination of an association.

6.3.4 Services Used within an Association

Bind-ACI-to-Message (Data)

An association security context can permit the transfer of data with varying ACI. This service lets a principal specify the ACI bound to data transferred within an association.

Message Protection Services

These services invoke cryptographic services to apply confidentiality or integrity protection to specific messages and data to provide protection during transmission within an association. The specific cryptographic mechanisms used to provide these services are determined as part of the security context established for the association. Examples of these services from the the **GSS-API Base** specification are *GSS_Sign*( ) and *GSS_Verify*( ) (which provide data origin authentication and data integrity protection), and *GSS_Seal*( ) and *GSS_Unseal*( ) (which in addition to the services supported by *GSS_Sign*( ) and *GSS_Verify*( ) provide support for confidentiality protection).

Note: The concept of quality of protection applied to operations within associations represents a trade off of performance against strength of security mechanism.

6.3.5 Impact on Primary Service APIs

Figure 6-5 illustrates a caller unaware of the secure association.

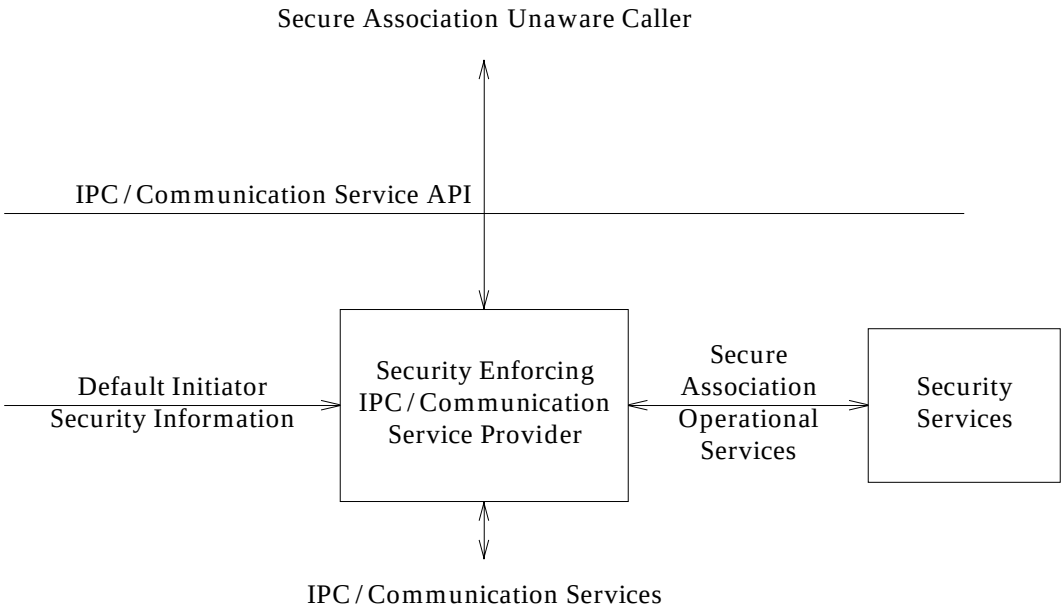

Figure 6-5 Secure Association Unaware Caller

A secure association unaware caller need not assume responsibility for any aspect of security association policy enforcement. The security policy is enforced entirely by the primary service provider and the API presented by the primary service provider need not provide such support.

A security enforcing provider requires that the default operational security context (initiator's credentials) within which the primary service is invoked has been previously established. That is, the initiator ACI has been generated and made available to the primary service provider either by inheritance from a previous process or from previous actions of the current process. In addition, the required Quality of Protection (QOP) has been previously defined, either by management action as a default or by a previous QOP security aware caller.

Figure 6-6 on page 119 illustrates a Quality of Protection Selecting Caller.

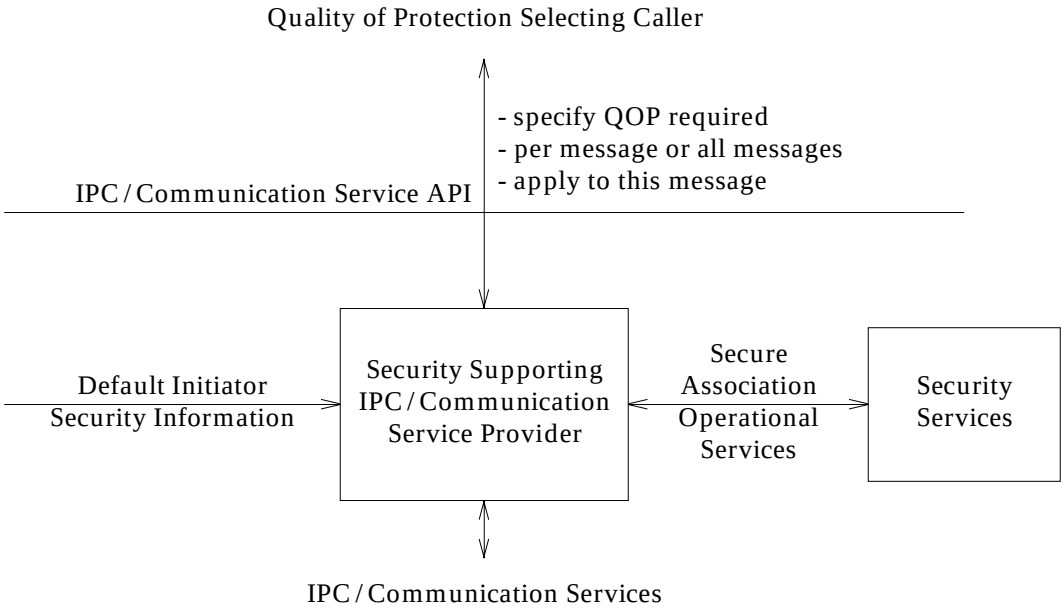

Figure 6-6 Quality of Protection Selecting Caller

A security-supporting primary service provider that supports QOP selecting callers must provide an API that permits the caller to specify the QOP for information exchanged by means of an association. This support may be on a per association basis or a per message basis or both.

The caller is not concerned with how the requested QOP is implemented.

Figure 6-7 on page 120 illustrates a security context selecting caller.

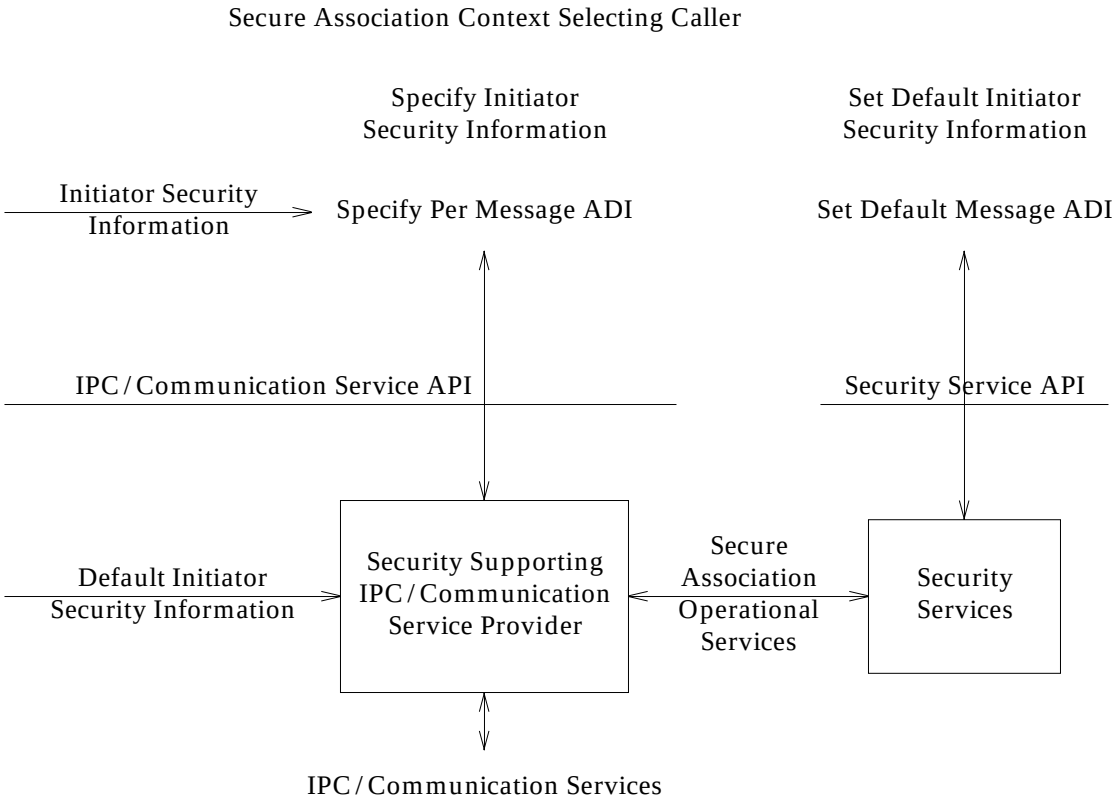

Figure 6-7 Security Context Selecting Caller

To allow for the selection of a security context, security supporting primary service providers must enable the caller to specify the security context for the creation of an association or the binding of security information to information exchanged within an association. A security context selecting caller is responsible for managing the initiator credentials used in the establishment of the security context of a secure association or for the specification of the ACI bound with data exchanged within the association.

If a security context specification service is not provided by a primary service provider a security context selecting application must use out-of-band methods to control the security context. It can do this by directly invoking interfaces that control the default security context of the primary service.

Figure 6-8 on page 121 illustrates a secure association enforcing caller.

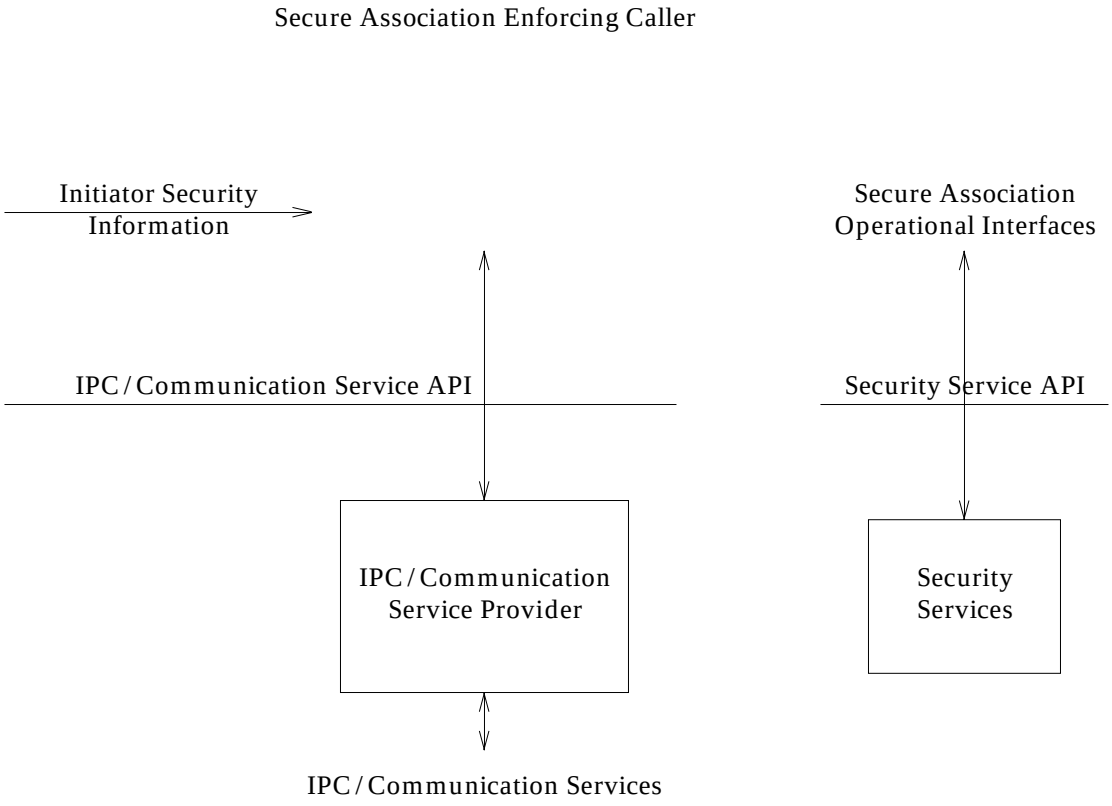

Figure 6-8 Secure Association Enforcing Caller

A secure association enforcing caller is responsible for the invocation of the secure association services. The primary service provider need not invoke the secure association service nor provide any support for the specification of the security context.

6.3.6 Network Security Services

Figure 6-9 illustrates a possible relationship between the service domain security services and the services supporting the security protocols within the transport and network communication service layers. The Network layer security protocol and Transport layer security protocol need an established security context that specifies the quality of protection applied to packets and the algorithms and keys used.

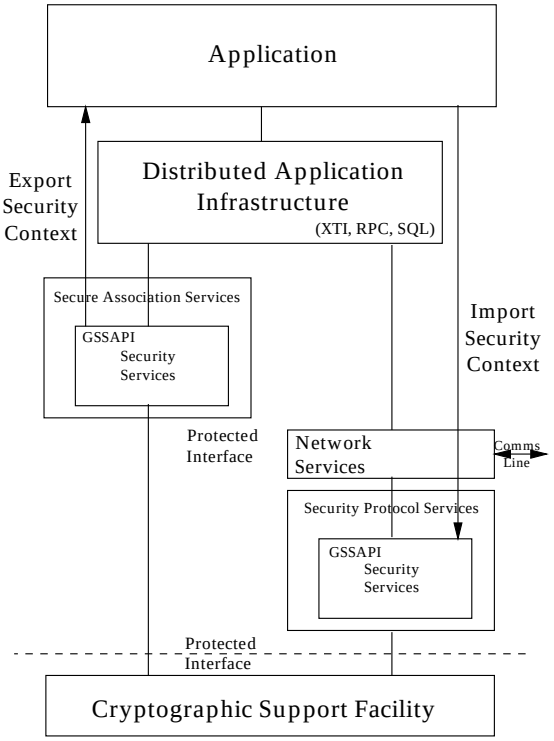

Figure 6-9 Service Domain and Network Security Services

Generally an application is not able to pass a handle to a security context to the communication services as they could be in a different address space; in that case a security context handle is meaningless. The export and import security context operations identified as secure association operational services must allow the security context to be passed.

6.4 Operational Information Acquisition and Verification

These services underlie the User Sign-on and Secure Association Service and the details are typically mechanism specific. They are not expected to be invoked directly by applications.

These services provide for:

- the acquisition of a set of operational security information, a security context, and its encapsulation as a security token
- the verification of security tokens submitted to it.

These services provide an interface to the services of a set of security servers. These security services include privilege attribute service, key management services and certification authority. This service also hides the details of cryptographic-related services from applications.

6.4.1 Management Services

These manage the configuration of the security services that can be used by the acquisition and verification service. The terminology used here does not imply only on-line servers but could also refer to the configuration of off-line servers.

Install-Security-Server

This service adds a security server to the underlying security service to the acquisition and verification service.

Modify-Security-Server

This service modifies the configuration of a security server within the acquisition and verification service.

List-Security-Servers

This service lists the details of the security servers configured for use by the acquisition and verification service.

Deinstall-Security-Server

This removes a security server from the list of those available for the acquisition and verification service.

Disable-Security-Server

This service prevents the acquisition and verification service from using the specified security server.

Enable-Security-Server

This service allows the acquisition and verification service to use the specified security server.

6.4.2 Operational Services

The acquisition and verification operational services comprise:

Acquire-Operational-Security-Information

This service retrieves a set of operational security information referenced by the specified credential handle and returns it to the caller. The operational security information can be protected by encapsulation as a security token for return to the caller.

If modification of the set of operational security information is requested (for example, for delegation purposes) a privilege attribute service can be invoked.

Verify-Operational-Security-Information

This service verifies the data origin and integrity of the specified security token. This could include the verification of security certificates or a chain of such certificates in the case of interconnection between security domains using different security servers.

When the security information is stored or propagated by a domain trusted by the target domain to maintain the integrity of the information, this service can map to a null service. An example is when a service domain trusts a platform domain storing security information as process attributes to maintain their integrity.

6.4.3 Impact on Primary Service APIs

The acquisition and verification operational services are not invoked directly by applications but are used indirectly by means of the secure association service.

6.5 Privilege Attribute Service

This service supports the acquisition of operational security information. The term Privilege Attribute Service is used in preference to Operational Security Information Service because it is in common usage. The Privilege Attribute Service provides authorisation operational security information such as ACI and ADI for initiators, targets and actions. Usually it also provides operational security information related to other security services, for example, for audit an audit ID and audit event discrimination information.

The callers of the Privilege Attribute Service are typically sponsor services such as:

User Sign-on Services
These sponsor users as principals into the information system domain and support user control over their security context, for example changing role or group.

Association Management Services
These sponsor a principal active within one service domain to another service domain by means of the Secure Association Service. This may include the facility for delegation of privileges to the other service domain.

Note: The Privilege Attribute Service could support the provision of other contextual information to the sponsor in addition to security information, for example, cultural information such as locale, but this is not considered further in this document.

Operational security attributes are typically issued as a Privilege Attribute Certificate (PAC). This is a security certificate signed by the Privilege Attribute Service which permits its origin and validity to be verified by any service domain to which it is submitted. The PAC may therefore be used to assert a principal's privileges and attributes under other security services represented within the PAC, for example, identity for accountability.

The privileges asserted by a PAC can be qualified by the holder to support the delegation of a principal's privileges. The delegated privileges would be for use by other principals on the initiating principal's behalf. In the context of this framework the other principals are service domains providing a specific service, for example, printing or a RDBMS.

6.5.1 Management Services

The management-related operational information services are as follows.

Install-Entity-Security-Attributes

This service installs a set of security attributes for one of the following:

a principal
The security attributes for a principal would normally be installed as part of user account installation.

In addition to security attributes for individual principals, it would be possible to configure default security attributes applicable to all or sets of principals.

a sign-on location
A sign-on location could be a terminal or workstation. The security attributes of a sign-on location would typically be used to provide contextual information and could also be used to qualify the privileges granted to a user as principal. For example, in a system that supports sensitivity labels, the clearance assigned to a sign-on location is used to constrain the clearance assigned to a user session.

In addition to security attributes for individual sign-on locations, it might be possible to configure default security attributes applicable to all or sets of sign-on locations.

The security policy of the privilege attribute service might require the authentication of the sign-on location as well as the principal.

a platform (or end-system)
The security attributes of a platform would typically be used to:

- provide contextual information
- constrain the security attributes included in a PAC
- support the control of the delegation of privileges by restricting a PAC to a particular end-system or set of end-systems.

In addition to security attributes for individual platforms, it might be possible to configure default security attributes applicable to all or sets of platforms.

a service domain (or application)
The security attributes of a service domain would typically be used to support control of delegation of privileges to a particular service domain or set of service domains. A service domain can be identified by both the application it supports and its platform location.

Modify-Entity-Security-Attributes

This service modifies a set of previously installed security attributes for an entity.

Revoke-Entity-Security-Attributes

This service revokes a security attribute or set of security attributes for an entity. This service differs from modify in that it implies that any operational security attributes currently issued by the Privilege Attribute Service and in use are revoked. This requires notification to the sponsor services to take appropriate action for any operations for which those security attributes are being used. For example, if ADI is revoked, any PAC including such ADI might have to be invalidated.

List-Entity-Security-Attributes

This service returns the values of all security attributes assigned to an entity within the Privilege Attribute SMIB.

Map-Security-Attribute-Value-to-Text-String

This service returns a textual representation of a supplied security attribute value for a user.

Map-Text-String-to-Security-Attribute-Value

This service returns a security attribute value for a supplied textual representation.

Configure-Privilege-Attribute-Security-Policy

This service supports the configuration of the security policy governing the privilege attribute service. For instance, it might include defaults for aspects such as PAC lifetimes, and establish trust relationships with authentication servers and other privilege attribute servers.

Disable-Privilege-Attribute-Service

This service disables a particular instance of a Privilege Attribute Service and prevents it from issuing operational security attributes.

Enable-Privilege-Attribute-Service

This service enables a particular instance of a Privilege Attribute Service and allows it to issue operational security attributes.

6.5.2 Operational Services

The operational-related Privilege Attribute services are as follows.

Acquire-Initiator-Security-Attributes

This service allows a caller to acquire a set of initiator security attributes; these can then be used to assert the initiator's privileges within service domains that recognise the authority of the Privilege Attribute Server. The initiator must assert its authenticated identity (Exchange AI) or an existing set of privileges represented by a PAC (ADI) to obtain the service. The authority of the security servers that issued the Exchange AI or the PAC must be recognised by the current Privilege Attribute Server. This means there must be a trust relationship between them.

Acquire-Delegate-Security-Attributes

This service allows a caller to acquire a set of privilege attributes which can be passed to another principal to assert a limited set of the caller's privileges. The security attributes may be restricted to use by a specific service domain on a specific platform.

Release-Operational-Security-Information

This service notifies a privilege attribute service that a set of operational information it previously issued is no longer in use, for example, on user session termination.

6.5.3 Impact on Primary Service APIs

The Privilege Attribute Service issues the operational security attributes providing the security context in which primary services operate. The PAC, a handle to a PAC, or information extracted from a PAC, can therefore be used as parameters to primary service APIs that provide specific support for callers that are security aware in some manner.

Primary services that support the creation of associations between service domains may have to support the specification of parameters used in calls to a privilege attribute service as a preliminary to the creation of the association. An example is controlling the delegation of the privileges of the caller to another principal.

6.6 Interdomain Service

For security domains to interact, they must exchange security information under the auspices of a secure interaction policy. There are different cases regarding the interpretation of the security information exchanged:

- The semantics and representation of the information and policy in both domains are identical. In this case translation is unnecessary.
- The semantics of the information and policy are identical, but not their representation. In this case a syntax translation is necessary.

  For example, two systems may both use uids to identify users and represent their access privileges but a user may be allocated two different uid values on each system; it is then necessary to map a uid value on an initiating system to a uid value on the target system.
- The semantics and the representation of the rules of each of the security domains are different. This means that an agreed set of rules needs to specify how the security information and policy of one domain can be translated into the security information and policy of the other domain because the description method is different and syntax translation is necessary.

  A simple example is interactions between a security domain that supports access control service and a security domain that does not. This requires the security domain that supports access control to assign an appropriate set of access control information to all data exchanged with the security domain that does not support access control.

6.6.1 Management Services

The interdomain management related services are as follows:

Install-Interdomain-Mapping

This service installs a mapping between the representation of operational security information within the host security domain and the representation of the operational security information within another security domain. The other security domain might be represented by a secure interaction security domain that has a different, but commonly accepted representation of operation security information that is only used for the purposes of exchange. Each of the interconnecting security domains then maps to this common exchange representation. An example of this is the token format used by TSIX(RE).

This could include the assignment of default values for some operational security information applied when the other domain does not support a security service supported within the host security domain. An example is the assignment of a default sensitivity label to data received from a system that does not support a label-based security policy.

Modify-Interdomain-Mapping

This service modifies an interdomain mapping previously installed.

Delete-Interdomain-Mapping

This deletes an interdomain mapping previously installed.

Disable-Interdomain-Service

This service places an interdomain service in a state in which its services are not available for use. This may be needed, for example, during the installation, modification or deletion of

interdomain mappings for the purposes of security policy integrity.

Enable-Interdomain-Service

This service enables an interdomain service and makes its services available for use.

6.6.2 Operational Services

The interdomain operational-related services are invoked as part of the secure association service which therefore protects applications from the details of this service. This is illustrated in Figure 6-10.

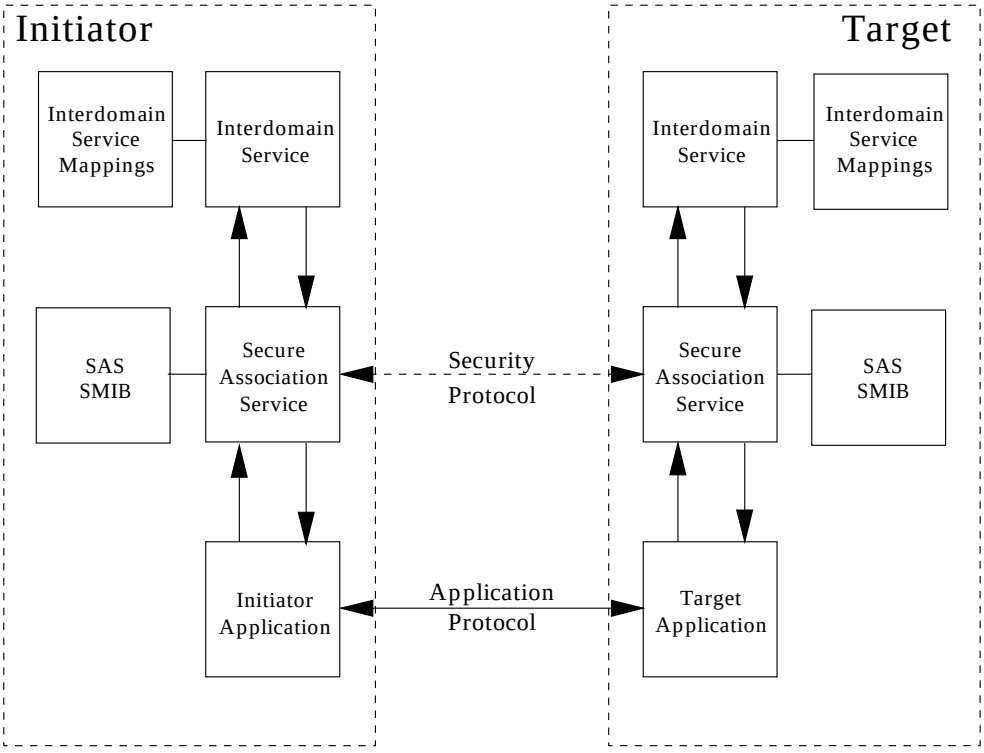

Figure 6-10 Interdomain Service

6.7 Key Management

Key management services include the generation, storage, distribution, deletion, archiving and application of cryptographic keys in accordance with a security policy. In performing these functions the key management services utilise the key management support services of the Cryptographic Support Facility.

6.7.1 Key Distribution

The purpose of key distribution is to transmit cryptographic keys securely between principals that might be located on different end-systems.

For situations where cryptographic techniques must be used to set up and maintain secure communication, a key-distribution mechanism is involved as part of the process of establishing a secure association between communicating peers.

Different mechanisms for distributing keys exist. These include:

- symmetric
- asymmetric
- manual.

Key distribution service providers typically utilise the underlying services of the Cryptographic Support Facility.

The operational and management services defined for the secure association service insulate the users of key distribution services from the details of the mechanism. The relationship of Key Distribution services to applications and the Cryptographic Support Facility is illustrated in Figure 5-7 on page 68.

6.7.2 Public Key Management Services

Domain interaction security service applications (such as operational information provider services, key distribution services or trusted third parties) or any security aware application might need access to public key management services to:

- hold asymmetric key pairs
- require access to the certificates used to bind the identity associated with each public key.

A public key management (PKM) service provider encapsulates the mechanisms used to manage public keys and their certificates. This includes all steps from generation of an asymmetric key pair to its final expiry or revocation.

The PKM comprises the following:

- an operational interface providing access to certificate and key handling services
- a Certification Authority
- an administration service used to generate, certify and revoke key pairs.

The PKM services are as follows.

Get-Public-Key

Returns a public key for a specified principal, and ensures that the validity of the public key is verified (using appropriate certificates, uncertified public keys, revocation lists, and so on). The caller can optionally specify a certificate identity; then the related certificate is guaranteed to have been used when verifying the key.

Get-Private-Key

With the same inputs as Get-Public Key, this service does the same checks, but instead returns the private key.

Query-Certificates

This is used to find out which certificates are involved in validating a public key. Security Aware applications can utilise this additional information to select a public key from a number of possible alternatives.

Check-Validity

This is used to check the remaining validity time for a public key.

Store-Certificate

This adds the certificate supplied for storage in the most global storage area accessible.

Put-Uncertified-Public-Key

This installs an uncertified public key into PKM. This results in the underlying Cryptographic Support Facility being invoked to initialise and store the key using a mechanism appropriate to the algorithm.

Put-Uncertified-Key-Pair

This installs an uncertified public key and its corresponding private key into PKM.

Delete-Key

This removes a public key and its corresponding private key from PKM storage.

Get-Other-Key

Given one of the keys of an asymmetric key pair, this returns the other if it is available and also accessible by the caller.

6.8 Certification Authority

The function of a Certification Authority (CA) is:

- to produce certificates (for example, as defined in the X.509) for public keys presented to it by bona-fide key holders
- to produce revocation lists as requested and according to policy.
- to satisfy itself as to the validity of a request before actioning the request containing it.
- to certify principals' public keys (which includes the public keys of other CAs, thus enabling hierarchies of CAs to be established)
- to maintain and publish a list of revoked certificates.

There can be many CAs, and one CA can serve several domains. The CA is typically off line, but a registration authority (RA) component can act as an on-line front end to the CA. The function of the RA is to receive requests and dispatch them to the CA for action, then return the appropriate response to the initiator

The CA may be used to certify the public keys of on-line security services such as an Operational Information Provider Service and Key Distribution Service. It could be also used to certify the long-term public keys of any other principle, for example an interdomain service, or security aware application.

A given CA can serve several domains. To allow domains under different CAs to interwork, domains can exchange uncertified public keys. Alternatively, a given key can be certified by more than one CA. CA hierarchies and cross-certification can be used as determined by policy.

6.8.1 Certification Authority Services

The CA and RA services are as follows.

Create-Certification

Create a certificate binding the specified identity and public key.

Revoke-Certification

Revoke the certification, and hence the identity specified by the certificate.

Cross-Certify

Create a certificate binding the specified CA and public key.

6.8.2 Certification Services

The Certification services are as follows:

Get-Certificate

Retrieve a certificate (locally or from a directory).

Initiate-Certification-Path-Construction-Context

Set up a context for constructing a certification path.

Construct-Certification-Path

Construct a certification path for the specified domain. If Construct-Certification-Path is called, but it has insufficient information to complete the path, it returns the name of a key holder or CA for which the caller should obtain certificates.

Terminate-Certification-Path-Construction-Context

Terminate a certification path construction context.

The public key management service provider uses the certification path construction functions to construct a certification path. These functions interrogate the trust criteria set up using the trust-path functions when deciding how to construct a path. All these functions work both with a flat certification model and a hierarchical certification structure with cross certificates.

6.9 Trusted Third Party Services

This section is based on the description of the non-repudiation service within ISO/IEC 10181-4. The working group has insufficient experience of the use of these types of services to address this area in any greater detail. Future releases of this document may add further detail to these services and consider their impact on other services.

The non-repudiation service involves the generation, verification and recording of evidence, and the subsequent retrieval and re-verification of this evidence should proof be required.

The purpose of the non-repudiation service described in this framework is to provide proof about a particular event or action.

6.9.1 Non-repudiation Service Model

The non-repudiation service model is illustrated in Figure 6-11 on page 135.

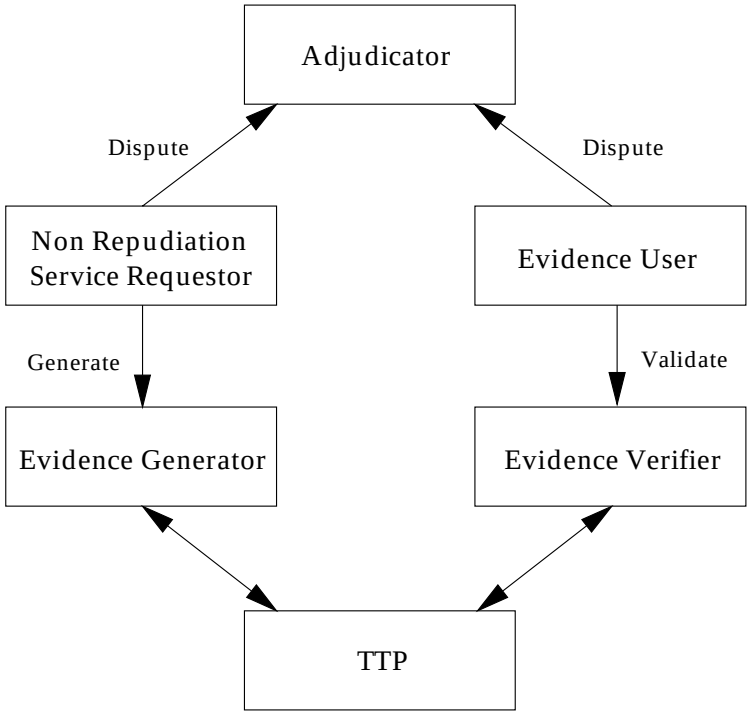

Figure 6-11 Non-repudiation Model

Non-repudiation can be divided into four phases:

- evidence generation — production of a record of a security event
- evidence transfer, storage and retrieval — subsequent handling and management of such evidential records
- evidence verification — verification of the occurrence of a security event
- dispute resolution — subsequent adjudication between parties on the authenticity of evidential records.

To support non-repudiation of an event a trust relationship is needed between the originator of an event and the recipient of an event. Such a trust relationship may be established through the use of asymmetric cryptographic techniques or trusted communication channels, or through the intervention of a trusted third party or parties.

6.9.2 Management Services

Management information for the non-repudiation service consists of those certificates, keys and trust dependencies required to support the operation of the non-repudiation service.

Install

Install management information.

Modify

Modify management information.

Delete

Delete management information.

List

List management information.

The non-repudiation services may require support from a time service to ensure the application of appropriate timestamp information in an assured manner, and from directory, certificate and key management services.

6.9.3 Operational Services

Generate-Evidence

Generate-Evidence generates evidence of the occurrence of an event based on the:

- authenticated identity of the originator of the event
- authenticated identity of the requester of the non-repudiation service
- authenticated identity of any intermediaries in the formation of the trust relationship leading to the occurrence of the event
- authenticated identity of any intermediaries in the formation of the trust relationship including a certified timestamp [this clause (qualifying trust relationship) deleted: leading to the propagation of the evidence to the requester state of the system at the time of the event]

- information on the nature of the event itself.

Validate-Evidence

Validate-Evidence accepts evidence generated by the above service and returns an indication of the validity of such evidence given the expected state of the event and the environment within which such an event occurs.

Assure-Evidence

Assure-Evidence provides additional assurance by requesting that a trusted third party provide a statement regarding an event, or that such a third party endorse an existing evidential statement.

Deposit-Evidence

Deposit-Evidence deposits an evidential statement with a trusted third party for later retrieval during arbitration.

Retrieve-Evidence

Retrieve-Evidence is used to retrieve an evidential statement that has been previously deposited with a third party.

Chapter 7

Security in Open System Services

This chapter provides guidance to developers of APIs on the security features that should be incorporated in their designs. Section 7.4 on page 139 to Section 7.14 on page 143 inclusive identify security requirements that must be considered by the specifiers of major functional areas.

7.1 Impact on API Specification

The primary service provider is responsible for invoking the security services using the appropriate operational security information by means of the security operational API.

All invocations of an API are operations on behalf of a principal. The operations are subject to the security policy rules of the security domain within which the API operates. The specification of a primary service must describe the security policy to which the service is subject. It must also specify the responsibilities of each of the basic security services in support of that security policy, such as:

- exchange and propagation of security information
- security certificate validation
- support for data origin authentication
- protection of data in transit or storage, for confidentiality or integrity
- access control decision or enforcement
- audit of events.

The security policy must be expressed in terms of the data and operations of the service itself, not in terms of a mapping to data outside the realm of the service.

All invocations of an API occur within a security context. This context is represented by security information associated with the initiator (caller of the API), the targets (data accessed by the API) and the operation itself. To specify a primary service, first identify the security information that comprises the security context of the service, and then identify how that security information is stored, propagated and retrieved.

To help with this procedure, an architectural model of a service should be produced, showing the interaction of components of the service, and how they map to platform data. The security attributes can then be assigned to the components, together with appropriate responsibility for the propagation of security information, and the enforcement of the security policy defined for the service.

Assertions can be made on the security properties of the infrastructure, the platform domain supporting the service and any other service domains used. An example assertion is that another service domain (login) has responsibility for the assignment of values representing the privilege attributes of an authorised principal. This relieves the primary service of any responsibility for the authentication of principals.

The API designer must decide whether the API should allow its caller to specify any information concerning the security context of the operation.

7.2 General Approach

The general approach to the specification and design of security mechanisms and measures within a functional area is summarised below. It is based on the principles, objectives and strategies introduced in earlier chapters. This work must be conducted within the general constraints and objectives of the framework.

1. Adopt the concepts of Chapter 3. View the functional area as one or more security domains with dependencies and interactions amongst themselves and the security domains representing other functional areas.
2. Perform a threat and vulnerability analysis for the functional area using the list of threats and vulnerabilities outlined in Appendix A.
3. Identify security measures and map these services within the functional area; establish which services enforce security and which support security with respect to each security measure.
4. Define interfaces to the security-relevant services that support security aware and security unaware callers as defined in Chapter 4.

7.3 General Functional Requirements

The security requirements common to most functional areas are:

Access Control

This is the provision of services that meet object reuse requirements by purging redundant service objects of all information.

Availability

This is general compliance with the principles of secure shutdown and recovery.

Accounting and Audit

This is the provision of audit event detection functions.

7.3.1 Language Services

To the extent that a programming language provides access to services within other functional areas, it also utilises the security features incorporated in those services. There is no need to levy additional requirements on language services.

Certain features of a programming language allow programmers to write more secure programs. For example, data abstraction can be used to hide details of security-relevant processes. As another example, the ability to write self-documenting code makes it less likely that malicious code could be hidden in a program.

7.4 System Services

The main security requirements on the operating system level are discussed in the preceding descriptions of platform domains. They are summarised as follows:

- Provide low level elements from which the higher-level services can be substantiated and on which the protected and separated service domains can be formed.
- Mediate access to all operating system elements according to any one or combination of the appropriate and required access control strategies.

Based on these fundamental security services, the operating system might also be required to provide specific elements to perform or support.

Authentication

The operating system should provide:

- secure storage of and access to authentication credentials
- support for a trusted path between user interface devices controlled by the operating system and higher-level authentication services.

Access Control

In addition to the basic operating system access control services the operating system may need to provide:

- secure storage of and access to crypto-variables (keys) and crypto-services
- secure storage of and access to the security context information (authentication and authorisation information) associated with each chain of execution passing through the environment controlled by the platform domain
- secure storage and access to security-relevant object information on behalf of itself or higher-level services
- support for the combination of authorisations to perform *n*-person operations.

Accounting and Audit

The operating system should provide:

- service functional elements for: security audit event detection for operating system functions, recording, alarm reporting, and the setting of audit and alarm selection criteria for operating system events
- secure storage and management of audit trails.

Administration

The operating system should provide: low-level service elements used to maintain security-relevant operating system management information. Examples are access control lists for system, user interface, storage, network and application software services.

Miscellaneous

The operating system should provide translation services that map security-relevant information from one representational or semantic form to another.

7.5 Communication Services

This mapping of security requirements and obligations on communication services does not yet take the mapping down to specific communication service layers.

Authentication

The communication service should provide service elements and protocols to support authentication between peer principals.

Access Control

The communication service should provide:

- confidentiality and integrity protection services to higher-level service elements which counter the transmission threats and vulnerabilities
- secure storage of and access to crypto-variables (keys) and crypto-services, when not provided by some specific service.

Accounting and Audit

The communication service should provide the functional elements responsible for countering communication repudiation threats.

Administration

The communication service should provide service elements used to maintain security-relevant communication service management information.

Miscellaneous

The communication service should provide translation services that map security-relevant communication information from one representational or semantic form to another.

7.6 Distributed Processing Support Services

Authentication

The distributed processing support services should provide:

- service elements and protocols to support authentication between instances of distributed processing support services
- authentication services which securely pass authentication credentials between the elements of higher-level services.

Access Control

The distributed processing support services should provide:

- secure association services which securely pass the security context between elements of higher-level services
- confidentiality and integrity protection service elements which counter the transmission threats and vulnerabilities
- secure storage of and access to crypto-variables (keys) and crypto-services, when not provided by some specific service.

Administration

The distributed processing support services should provide service elements used to maintain security-relevant distributed processing support service management information.

Miscellaneous

The distributed processing support services should provide translation services which map security-relevant communication information from one representational or semantic form to another.

7.7 Database Services

Authentication

The database services should provide or utilise provided authentication services to authenticate the identities of principals invoking the database services.

Access Control

The database services should:

- assign privilege attributes to authenticated principals, or verify the authenticity of privilege attributes presented to it
- bind access control information to elements of the database (tables, records, and so on)
- bind the privilege attributes of the principal with the set of operations performed on its behalf by the database services
- mediate access to the database elements and operations according to the required access control strategies.

Accounting and Audit

The database services should provide for audit event detection for database operations and providing, or interfacing to provided, services for event recording and alarm reporting.

Administration

The databases services should provide services to maintain database security management information. For example, access control lists associated with the elements of the database.

7.8 Data Interchange Services

Data description protocols and data format protocols are defined at a level that may not require security functions. Instead, security is wrapped around data encoded in those formats at the time of exchange. However, particular data formats may need to include authentication information.

7.9 Transaction Processing

The security requirements for transaction processing are subsumed by the database, operating system or communication services under which the transactions occur.

7.10 User Command Interface

The security requirements for the user command interface are utilities for user and shell-application control of access control lists and other aspects of the security interface (see the ISO POSIX-2 standard and the draft POSIX.1e amendments).

7.11 Character-based User Interface

Authentication

The character-based user interface should provide:

- service elements and protocols to exchange securely authentication credentials between the user and the system authentication services
- support for trusted path mechanisms.

7.12 Graphical Window System

Authentication

The graphical window system should provide service elements and protocols to exchange securely authentication credentials between the user and the system authentication services.

Access Control

If the window system allows simultaneous display of windows in different security domains, the window system must maintain separation between these domains. For example, cut and paste functions must be aware of what domains they are moving data between, and must disallow such movement as security policy requires. This may in turn require management functions to specify the policy. Another example is visual labelling of windows, indicating the security domain displayed, as an aid to the user in maintaining separation.

7.13 Graphics

Not applicable.

7.14 Application Software Development

Configuration management must ensure that malicious code does not get incorporated into developed programs. Any tools that improve the quality of code developed are security-enhancing, since they reduce the chance of code having security vulnerabilities.

Appendix A

Threats and Vulnerabilities

This appendix expands on the generic security threats introduced in Chapter 3:

- unauthorised modification
- unauthorised disclosure
- unauthorised use of resources
- denial of service
- repudiation.

These threats are examined with respect to security vulnerabilities and methods of attack. The objective is to acquaint the reader with the areas of concern and the interdependent nature of the threats.

The term security vulnerabilities refers to the weaknesses in the construction, capability and operation of information systems that expose them to the accidental or intentional realisation of security threats.

The intentional exploitation of information system vulnerabilities is known as a method of attack. The methods of attack used may vary according to the nature and capability of the system and its components as well as the objectives and skill of the attacker.

The following analysis mainly concentrates on the vulnerabilities and methods of attack that generally fall within the scope of IT to address.

A.1 Unauthorised Modification

With respect to unauthorised modification there are two areas of concern and vulnerability:

- modification of software and hardware
- modification of user, application and system data.

A.1.1 System Software and Hardware Modification

The absence of, or poor system configuration management facilitates the introduction of unauthorised software and hardware components. Such components can contain malicious capability that can damage the integrity of the system. Examples of such attacks are described in the following sections.

Introduction of Malicious Software

The common forms of this are:

- *Virus software*: this can propagate itself; it often carrys a payload designed to perform some malicious act.
- *Trojan horse* components: these partially or fully emulate security-relevant software components to capture and relay security-sensitive information.
- Security and business-relevant software components containing *trapdoor* capability: this permits those who know about it to bypass some part of its normal operation. This category includes the wide range of flaws and undocumented features common to much software.
- Software components designed to perpetrate some malicious or fraudulent act either at some specific time or when signalled to do so: these are sometimes referred to as *logic bombs*. They may be inserted as part of a virus, or be specifically engineered, usually by someone with detailed knowledge of the target system.

Introduction of Unapproved Hardware

Unapproved or unchecked hardware can contain hardware and firmware features that allow existing security mechanisms to be bypassed.

A.1.2 Data Modification

The preservation of the integrity of enterprise-sensitive and critical information is usually based on the concept of well formed transactions. Such well formed transaction models specify the stages in the life cycle of each type of enterprise information. For each stage, whether it be information creation, modification, transmission or destruction, the well-formed transaction model should capture and define the authorisations and cross checks required to perform the stage.

Vulnerabilities and methods of attack on the integrity of system, user and application data depend on the completeness and correctness of the appropriate transaction models as well as their IT implementation.

The weakness of and attacks on the integrity of system, user and application data are discussed in the following sections.

A.1.3 Transmission

Interception and Modification

Information communicated over networks can be intercepted and modified as it passes through weakly defended or uncontrolled links, routers and nodes. The classic example given is the modification, upwards, of the financial amount field of an authorised message requesting a credit to the attacker's account.

Insertion and Replay

Messages, possibly previously intercepted and modified, are introduced into the communication traffic either as part of a masquerade or fraud attack, or just to damage the integrity and availability of the communication services.

A.1.4 Storage

The integrity of stored data can be accidentally or intentionally compromised in a number of different ways. Many of these result from poor access control facilities. Access control facilities should govern not just who has access to what stored information but also what tools and utilities can be used to access the information. The more physical ways in which data storage media can be corrupted must be addressed by disaster planning and recovery, although software tools may be used to detect such physical corruption.

Specific Data Modification

As for transmitted data, specific stored data item values can be changed to support a fraud attack or to facilitate the operations of the attacker in some other endeavour.

Data Contamination and Corruption

Localised or general corruption of stored data may be either immediately obvious or only discovered after some time. The latter is more insidious as it could be copied into the entire backup archive.

Deletion

The unauthorised removal of sensitive stored data is possible.

Creation and Replacement

The unauthorised introduction or replacement of sensitive information is possible.

A.1.5 Processing

Malicious Software

The concern here is in the trustworthiness of the software components used to process sensitive information. Attackers may introduce their own versions of regularly used tools and application components (see **Introduction of Malicious Software** on page 146). These can be designed to modify specific instances of data to achieve any one of the range of objectives identified in this appendix.

A.1.6 Modification of System Data

The system and security administration data, and the services used to maintain this system data, determine the current operational configuration of the system. If not adequately protected, system and security management data is susceptible to all of the attacks already described; however, because it is critical to the general preservation of system security, it is explored further here.

There is a wide range of options available to the attacker as to how to exploit vulnerabilities in this area. Some of these are reasonably direct, obvious and detectable; others are more subtle, insidious and less easily detected. Examples are given in the following sections.

User Account Modification

Unauthorised access to user account or profile data and services can be used to:

- create unauthorised user accounts
- increase or decrease the privileges and authorisations assigned
- discover and use redundant user accounts
- remove active user accounts.

Communication Configuration Modification

Unauthorised access here can be used to create or modify communication addressing and routing parameters at different levels of communication service to open up new channels and even redirect communication.

Changing Clocks

There are an increasing number of security mechanisms that are dependent on correct time synchronisation. An attacker who manages to falsify a system clock may well succeed in defeating these mechanisms.

System Service Modification

More generally, inadequately controlled access to system administration and management services can be exploited by an attacker to perform disclosure, modification and denial of service acts.

A.2 Unauthorised Disclosure

There is a variety of ways in which the confidentiality and privacy of information within information systems can be compromised. The vulnerabilities to unauthorised disclosure threats occur in the following areas.

A.2.1 Transmission

Eavesdropping

Sensitive information transmitted over local and particularly wide area networks is liable to eavesdropping at diverse points in the layered communication services and networks. The term *sniffer* is used to cover a component designed for eavesdropping. Typically a sniffer is placed at some strategic point in the communication services from where it can observe and report sensitive transmitted information, for example user ids and passwords.

Traffic Analysis

In certain cases the frequency, volume, origin and destination of messages is considered to be confidential information. An attacker may discover such information through the analysis of communication traffic flow.

A.2.2 Storage

Copying

Information can be subject to unauthorised copying using any one of the many IT techniques and tools for reading and producing copies of stored information.

Garbage Searching

A sometimes overlooked vulnerability arises in the tendency within all levels of IT system services to make transient copies of information as it is processed and transmitted. These copies often take the form of large internal buffers or temporary files occupying reusable *garbage space*. Thus an attacker may use the electronic equivalent of garbage collection and analysis.

A.2.3 Processing

Malicious Software

As for unauthorised modification, attackers can introduce their own versions of regularly used tools and application components. These can be designed to capture and relay the sensitive information they process.

Electromagnetic Radiation

If not adequately shielded, computers, terminals and electronic communication equipment radiate electromagnetic signals which can be picked up at distances varying according to the strength of the signal. With the appropriate equipment, the displayed, processed or transmitted information can be reconstructed.

A.2.4 Indirect Disclosure Vulnerabilities and Attacks

There are more derived classes of confidential and private information subject to indirect and subtle forms of attack. Examples are given here to ensure that they are included in the future development of X/Open security guidelines.

Aggregation

Sensitive information can be deduced through the assembly of less sensitive information. For example, this could be achieved by the unauthorised construction of a detailed description of an individual by aggregating the parts of the description from different sources.

Deduction

The fact that certain data is being held on a system or database could be considered sensitive information. Although unauthorised direct access to this data could be effectively prohibited its presence can sometimes be deduced from the responses to carefully constructed searches.

Listening

Microphones and voice recording services are features recently introduced on some workstations. It appears that in certain cases the default configuration of these services permits other users, possibly remotely, to turn voice recording on without there being any visible indication to the owner of the workstation. This disclosure vulnerability is included here to indicate the ever widening range of disclosure weaknesses in modern information systems.

A.3 Unauthorised Use of Resources

Unauthorised use in its widest context includes the theft of hardware and software resources. Although schemes and mechanisms for the electronic tagging of physical computer assets are emerging, addressing these is not currently within the scope of this document. Similarly, software licence management is outside the current scope.

Within scope are the vulnerabilities to the unauthorised use of systems and the services they offer. Weaknesses in the control of access to these resources can then lead to disclosure, modification, repudiation and denial of service attacks. The converse is also true, in particular, disclosure and modification vulnerabilities can facilitate unauthorised use attacks.

A.3.1 Discovery and Use of Authentication Credentials

Identification, but principally, authentication credentials are used to establish the authenticity of authorised users and other principals, such as systems and their component services, and even hardware. In many cases mutual authentication is needed to counter masquerade by either party.

With respect to principal authentication credentials, the main area of vulnerability lies in poor measures governing use, selection, renewal, and safe transmission and keeping of these credentials.

Where principal authentication credentials are used, the receipt, passing (sometimes over dubious long-haul networks) and storage of authentication credentials may be liable to the attacks described below. Discovered authentication credentials can then be used to masquerade as the principal by reusing the credentials or replaying them over the network.

Trojan Horse

This method of attack allows unauthorised collection, falsification or destruction of some part of the identification and authentication functions.

Eavesdropping

Interception is possible when information is passed over local and wide area networks.

Accessing the Credential Store

Discovery of authentication credentials is possible by access to their stored representation. Here, even though the credentials could be stored encrypted, poorly selected credentials can be liable to brute force decryption attacks.

A.3.2 Acquisition of Authorisations

As already identified in Section A.1.6 on page 148, an attacker can exploit weaknesses in the protection of authorisation information to acquire further authorisations and thus access services and information beyond that normally allowed to them. For example, in UNIX-like systems it is possible to acquire *root* or *superuser* authorisations.

A.3.3 Misuse of Access Rights

Legitimate users may use their authorised access rights to data to perform authorised transactions. An example of this is secondary discretionary disclosure.

A.4 Denial of Service

IT system vulnerabilities to denial of service and availability threats abound. They include areas of obvious concern such as:

- dependency on continuous power supply and communication links
- system hardware component reliability
- system software component reliability.

These are generally well recognised areas of system availability management and planning. The establishment of disaster strategies and plans, and the regular taking and securing of backup copies of sensitive information, is appreciated by most and practised by many. Apart from physical events and acts the accidental or intentional denial of service can be caused by:

Unreliable Software
This is the most common cause of denial of service.

Malicious Software
Many of the types of malicious software attack can be used to destroy or cripple a service.

Swamping
The availability of a system or service to authorised users can be severely reduced by flooding it with requests. Swamping may also overwhelm the system or service by exhausting the storage associated with it.

Of concern to IT system security are attacks that aim to disable some part of the existing security mechanisms. By taking advantage of the fact that on failure, either the system or some security-relevant service does not fully recover to a secure state, an attacker may bypass the normal defences.

A.5 Repudiation

Repudiation attacks are predicated on the existence of some measures for making users and other principals accountable for their business-sensitive or security-sensitive actions. There are two such general mechanisms:

Audit Logging
This is the recording of each action and the identity of the responsible principal in an audit trail.

Sealing and Signing
This is binding the identity of the responsible principal to information produced by the action.

The often quoted example of the latter case is the binding of digital signatures to electronic mail messages. In both cases additional information such as location, context and particularly time is usually recorded.

Attacks on such measures and mechanisms are levered off more basic attacks.

Masquerade
This is the use of the identity of another principal hide the true responsibility.

Audit Trail Modification
This is the breaching of the access controls to the audit trail and changing or deletion of all or only relevant audit records.

Malicious Software
This is the use of unauthorised software to perform the operation and to omit or falsify the record or signature.

A.6 Lack of Awareness and Utility

To put things in perspective, currently the most common vulnerabilities arise from the lack of awareness of information system security concerns, needs and practices by the users and owners.

There is often failure clearly to assign to users and administrators responsibility for information security. Addressing this falls outside the scope of this document. However, the concept of security domains, as described in Chapter 3, does provide a useful way of establishing boundaries of security responsibility and jurisdiction.

There are system security awareness and utility issues that do fall within the scope of the framework. They are summarised in the following sections.

Awareness

This is not knowing in sufficient time that a system has been attacked or that its security has been compromised. The vulnerabilities arise from inadequate measures for detecting and reporting attempted or successful attacks. Damage limitation and control is obviously dependent on timely detection and warning.

Utility

Even though strong security measures may have been deployed, if these are difficult to use and manage, their effectiveness is significantly diminished.

From the users' point of view, if the security behaviour of the system appears to impede normal operational efficiency, they are encouraged to find ways round the system's security.

The effectiveness and utility of system and security management are currently one of the greatest areas of vulnerability within distributed systems. Lack of facilities that allow large, heterogeneous distributed systems to be easily managed almost guarantees that they are riddled with vulnerabilities.

A.7 Assessment of Threats, Vulnerabilities and Risks

Security threat and risk assessment is typically an iterative process applied at each level: enterprise, IT system, IT sub-system and IT service. Simply, the assessment steps are:

1. Identify what assets must be protected against which threats.
2. Determine the vulnerabilities that can manifest these threats.
3. Estimate the risks of these vulnerabilities being exploited.
4. Where the risks are unacceptable, identify and specify a set of countermeasures to the threats with the aim of reducing the vulnerabilities and associated risks to acceptable levels.

The information system security concerns (risk-weighted threats and vulnerabilities) may vary from one enterprise to another. The proposal here is to use functional profiles to generalise these concerns.

However, in the X/Open specifications for each of the major functional areas of IT the list of threats, vulnerabilities and methods of attack, given above, should be interpreted and adapted to the context and capability of each area and component.

Appendix B

Availability

The concept of availability embraces the reliability, maintainability and survivability of systems to ensure the timely provision of services. The division of responsibility for assurance of availability between general system performance issues and security issues remains unresolved. It is clear that resistance to deliberate denial of service attacks is of security concern, although the mechanisms which ensure such resistance are also the mechanisms which provide resistance to accidental denial of service through component or subsystem failure.

Fundamental components of an availability architecture are:

- provision of a timely service through the use of components offering adequate performance to meet requirements
- ensuring that the service is not subject to interference by other services, and has access to all resources required to provide its service.

The architecture should also provide for graceful degradation in the face of component or sub-system failure. To assist in such degradation the availability architecture can make use of controlled redundancy, component reinitialisation and the use of alternative recovery strategies. These approaches are supported by the existence of fault and exception detection, reporting and analysis mechanisms.

B.1 Controlled Redundancy

This approach provides for the duplication of key components and sub-systems to ensure that appropriate redundant processing capacity is in place. Such systems include multi-processor architectures, RAID disk mirroring strategies, alternative communication channels and data distribution strategies. The management of redundancy includes the provision of API sets to support the indication of required quality of protection (QOP) for data storage, data transmission and data processing. Such generic interfaces are supplemented by interfaces to support initialisation of redundant system components and shutdown of failed system components. Examples are interfaces to support configuration of non-stop hardware modules.

Redundancy mechanisms are supported by fault detection mechanisms as noted above. Such mechanisms may be based on detection of component behavioural deviation from the expected profile (such as watchdog timer systems or pre- or post-condition analysis) or detection by majority voting mechanisms.

B.2 Component Reinitialisation

This approach assumes that transient errors are the major cause of system failure. There is a high probability of avoiding subsequent system failure if the failed components are reinitialised. Component reinitialisation varies from full system restart or reboot, through process restart to support for process rollback and transaction restart. Examples of interfaces to support component reinitialisation are:

- system reboot — an interface to provide for full system reinitialisation
- process restart — an interface to provide for reinitialisation of a given process or cluster of processes.

These interfaces allow for full reinitialisation of a specific system state. The system state can be updated through the provision of checkpoint and rollback mechanisms. Support is therefore required for checkpoint generation under process or external control, and the subsequent rollback to such checkpoints. This mechanism may be extended to offer a full transaction-oriented rollback mechanism with each transaction representing a secure and permissible system state transition.

In the absence of such transaction-oriented mechanisms, additional mechanisms are required to assess whether component reinitialisation has maintained the security of the system state. An example is assessment of whether damage to a file system is likely to allow unauthorised access to data. This implies the existence of secure recovery mechanisms including system state inspection.

B.3 Alternative Recovery Strategies

The final approach recognises that the system failures could be of sufficient regularity and consistency to prevent the application of reinitialisation or redundancy as recovery techniques. In such an event the system might be forced to adopt other recovery strategies offering reduced or alternative services. The management of such graceful degradation becomes critical, including assessment of the relative priorities of users in gaining access to the restricted services offered by the degraded system. Interfaces are therefore required to manage this degradation, including the provision for review of requested quality of services and the external modification of such requests in the light of system service priorities and policies.

B.4 Fault Management

To assist in the management of such recovery, reinitialisation and redundancy, it is critical that fault management mechanisms are in place. Some examples are given in the following sections.

Fault Detection

The first requirement is the provision of appropriate mechanisms to ensure the adequate operation of system components. This may include the internal use of firewalls within systems such as *assertions* of procedure before and after fault conditions. External checks must be provided based on assessment of continued operation of systems or components, and the ability of users and administrators to indicate system failures and faults.

Interfaces could be required to indicate the required level of fault detection including the use of watchdog timers associated with key actions or events, and the specification of assertions on system state and behaviour.

Fault Monitoring

Interfaces are required to report faults to appropriate monitoring processes and subsequently to incorporate these reports in system logs and audit trails. The faults can be broken into per-system and per-process events. The former are asynchronous notifications of system-level failures, such as I/O errors and power failure warnings, which may not be directly related to a process service request. A mechanism is required to enable distribution of these events to relevant processes, with appropriate interfaces to allow processes to request access to event notifications. These interfaces could also handle the relative priority of access to such events, and be able to delete them from the system event queues.

Additional categories of events are directly associated with the actions of a process or processes including events such as floating-point numeric exceptions and segmentation violations. These have been traditionally implemented as UNIX signals. Extended interfaces are required to permit other processes to intercept these events and take action on behalf of other processes.

Interfaces are required to record fault events in system logs and to propagate these events to central nodes for additional processing and analysis.

Fault Analysis

Fault analysis mechanisms handle the subsequent reduction and assessment of fault events to determine causes, manage system degradation and provide operator feedback.

B.5 Key Interfaces

The primary areas in which APIs can be defined to support availability mechanisms are:

- augmentation of resource request APIs to include quality of service
- provision of checkpoint and rollback APIs to support transaction oriented processing and secure recovery mechanisms
- provision of APIs to support the interception and and reporting of fault events to processes
- provision of APIs to support the provision of timeouts associated with service requests and subsequent invocation of recovery action
- provision of extended process priority and scheduling mechanisms to support service segregation and prioritisation of access
- provisions of APIs to enforce pre-emption of resources.

Appendix C

Example Mappings to Framework

This appendix presents interpretations of a number of common system types in terms of the the security framework. These are summarised in the table at the end of this appendix.

The services represented by the **XSH**, **Issue 4**, **Version 2** specification and the POSIX.1e Security Extensions currently being developed are mapped.

There is a mapping of the TSIX(RE) trusted IPC specification produced by the Trusted Systems Interoperability Group (TSIG).

C.1 Interpretation of ISO POSIX-1 and POSIX.1e

C.1.1 Security Domain

The System Service Security Domain is a Platform Domain within the context of this Security Framework. Within the Platform Domain the sponsor is a thread of execution within the operating system kernel; that means operations within kernel mode in a traditional (UNIX) implementation. The principals that use the services of the Platform Domain are threads of execution within Service Domains (applications), that is threads of execution operating within user mode in a traditional implementation.

The operations of the System Services Security Domain are invoked by means of the system service interfaces.

The elements of the System Service Security Domain are Operating System entities: processes, files, interprocess communication (IPC), and so on.

The system service boundary must be a protected interface that cannot be bypassed, if the system service security policy is to be enforced independently of the behaviour of any applications invoking its services. In a traditional implementation this is usually achieved with the support of hardware mode switching.

C.1.2 Principal Security Attributes

The security attributes of a principal, the thread of execution within a Service Domain (application), are bound to the thread of execution in the form of attributes of the process within which the thread is executing.

When a new process is created, by a *fork*( ) operation, these attributes are inherited from the parent.

The principal security attributes may be modified in two ways:

- By an appropriately privileged principal modifying the attributes itself in order to utilise the security services of the Platform Domain to enforce its own security policy and responsibilities. An example is a Sponsor Service Domain (for example, login, which sponsors a user within the system).
- On change of principal (change of thread of execution by an *exec*( ) operation) the Platform Domain may modify the principal security attributes on the basis of the setuid and setgid flags configured on the file to be executed. In this case, the change represents a change of Service Domain. In traditional terms this is the invocation of a protected subsystem. No authentication of the new principal is required as its identity and privilege attributes are already maintained by the platform domain.

C.1.3 Authorisation

An authorisation scheme based upon a restricted ACL scheme is supported by POSIX.1. The scheme is restricted in the sense that only three entries are permitted for owner class, group class, and other class.

In addition authorisation for some operations is based upon an unspecified appropriate privilege. In traditional systems this is often equated to a reserved uid, uid 0, termed *superuser* or *root*.

The majority of system service interfaces are of the security enforcing type. That is they do not provide for the specification of any security parameters but rely upon the system service implementation to utilise default security parameters (represented by process attributes). The

exceptions are those system services providing for the creation of some platform domain objects, for example files, in which file access permissions may be specified as part of the parameters to the system service interface. All other security attributes are derived as defaults from process attributes.

Some interfaces, for example *read*( )and *write*( ), enforce no authorisation scheme but assume that they are operating within the context of an established association in which the authorisation for the operation has been previously established as part of the association creation. The opening of a file is an example of the formation of such an association in which authorisation for the subsequent *read*( ) or *write*( ) operations is evaluated by the *open*( ) operation.

C.1.4 Mapping of System Interfaces to Framework Services

Propagate Privilege Attributes and Install Target ADI

fork( )
Creates a new process (branch) in the tree of execution. A new identity (process ID) is assigned by the platform domain. Other security attributes are inherited unchanged. Process attributes represent both the privilege attributes of the the principal (application thread of execution) and the target ADI of the process as a target of system service operations. This operation therefore both propagates privilege attributes and installs target ADI for the new process.

Install Target ADI

open( ), *creat*( ), *mkdir*( ), *mkfifo*( )
These all create new file system entities. A new identity (inode) is assigned by the Platform domain. Other security attributes are derived from the security attributes of the process from within which the service request is invoked. The exception is the file status flags and file access modes may be specified as a parameter to these calls.

Modify Principal Privilege Attributes

setuid( ), *setgid*( )
These interfaces permit an appropriately privileged process to modify the privilege attributes, initiator ADI, bound to the current branch of the tree of execution as attributes of the current process.

exec( )
This interface changes the process image associated with a tree of execution (changes the principal) and, if the setuid or setgid flags are set may also change the initiator ADI bound to the tree of execution.

Retrieve Principal Privilege Attributes

getuid( ), *geteuid*( ), *getgid*( ), *geteuid*( ), *getgroups*( )
These interfaces permit the initiator ADI bound to the tree of execution to be retrieved.

Set Default Target ADI

umask( )
Sets the default file access permissions mask.

List Target ADI

stat( ), *fstat*( )
These interfaces list attributes of files including the target ADI of files.

Modify Target ADI

chmod( ), *chown*( ), *chgrp*( )
These interfaces modify the target ADI of files. In addition *chmod*( ) permits the configuration of principal privilege attributes through the setting of setuid and setgid flags on an executable file.

Privilege Attribute Name to Id Mappings

getlogin( ), *getpwnam*( ), *getpwuid*( ), *getgrgid*( ), *getgrnam*( )
These are services that support the translation of platform security attributes (process attributes) into human readable forms and vice versa. A User Sponsor Service Domain assigns security attributes using the appropriate database. These services are applicable to the User Sponsor Service Domain rather than the Platform Domain.

Non-related Process IPC Security Services

P1003.1 does not include any IPC services for use between non-related processes. Therefore no security services have been defined for such services.

C.1.5 POSIX.1e Security Extensions to POSIX.1

POSIX.1e adds the following security mechanisms and services to POSIX.1:

- Additional authorisation mechanisms. These are an ACL scheme, a label-based scheme and a capability scheme.
- An informative label scheme is included. The objective of this scheme is to maintain the correct classification of the information contained within platform entities, such as processes and files. The authorisation schemes of POSIX.1 and POSIX.1e are restricted to the platform entities as containers. This scheme is not discussed further in this framework.
- A security audit option.

Access Control List Authorisation Scheme

The restricted access control list authorisation scheme is extended to permit more entries. A minimum number of entries is required to maintain compatibility with the POSIX.1 scheme.

An ACL is assigned to a newly created file on the basis of a default ACL assigned to the directory in which the file is created.

The privilege attributes of a principal (process attributes are unchanged by this addition.

The additional services are:

- Modify Target ADI — *acl_set_file*( ) and *acl_set_fd*( )
- List Target ADI — *acl_get_file*( ) and *acl_get_fd*( ).

Label Authorisation Scheme

Security labels are assigned to both principals (representing a security clearance) and to files (representing a security sensitivity). Authorisation is evaluated by comparing the security clearance of the initiator (process label) and the target (file label) according to implementation-defined rules of dominance and equality of labels.

The additional services are:

- Modify Privilege Attributes of Principal — *mac_set_proc*( )
- Retrieve Privilege Attributes of Principal — *mac_get_proc*( )
- Modify Target ADI — *mac_set_file*( ) and *mac_set_fd*( )
- List Target ADI — *mac_get_file*( ) and *mac_get_fd*( ).

In addition, services are provided to Service Domains to compare labels for dominance and equality. These are *mac_equal*( ) and *mac_dominate*( ).

Capability Authorisation Scheme

A set of capabilities are associated platform domain operations. Individual capabilities are associated with individual operations. A set of capabilities is assigned to each principal as represented by an executable program. On an *exec*( ) operation the capabilities represented in the privilege attributes of the principal (process attributes) are modified to represent the new principal. It is possible for a principal to manipulate the capability state of its privilege attributes.

The additional services are:

- Install Initiator ACI — *cap_set_file*( ) and *cap_set_fd*( )
- List Initiator ACI — *cap_get_file*( ) and *cap_get_fd*( )
- Modify principal privilege attributes — *cap_set_proc*( )
- Retrieve principal privilege attributes — *cap_get_proc*( ).

Security Audit

An audit id is added to a process attribute set. This value is set by a service domain component fulfilling the sponsor function (for example, login).

Security-relevant system service operations are required to audit their invocation.

An audit security service interface is supported to permit applications to create and submit their own audit records with the audit service.

A capability is associated with a system service operation to suspend auditing by the system service interfaces for the invoking process.

The security audit service therefore supports both security audit unaware callers and security audit generating callers. In the case of security unaware callers the system services are required to audit their invocation. Security audit generating callers are able to detect auditable events within their own operations, and then to generate audit records and submit them for recording using the *aud_write*( ) service. In addition, if assigned an appropriate capability, such a caller may use *aud_switch*( ) to suppress system service auditing of its operations.

C.2 Interpretation of TSIX(RE)

The referenced TSIX(RE) specification is a specification produced by the Trusted System Interoperability Group. The information detailed in this mapping is derived from the specifications available by means of anonymous ftp from ftp.sterling.com.

The TSIX(RE) specification comprises protocols and APIs which have been developed out of the Defense Intelligence Agency Compartmented Mode Workstation programme and related operations for the provision of network security services. It therefore has its roots in operating system TCB development and the interconnection of systems at the operating system level. The TSIX specification is expressed in terms of hosts and applications (or processes). Within this framework a host equates to a platform domain and an application to a service domain.

The current TSIX specification is intended to operate over a physically protected network on which all connected platform domains, and their respective system and security administrators, are trusted to support a minimum set of security requirements, see Section C.2.3 on page 167.

The TSIX(RE) specification therefore provides services at the Platform Domain level in support of interactions between Platform Domains and Service Domains hosted on those Platform Domains.

C.2.1 TSIX(RE) Services

The TSIX(RE) specification supports:

- Network Layer and Network Layer Management Services
- Session Management Services
- Message Security Attribute Services
- Interdomain Services and Management.

Network Layer and Network Layer Management Services

The network layer services comprise:

security attribute transport
: This is the binding of a set of security attributes with every Internet Protocol (IP) packet as an IP security label.

access control on the export and import of packets
: An access control policy is enforced on the transmission of packets between platform domains based upon the security attributes of the packet and the security attributes of the hosts.

trusted routing
: A host is able to seek alternative routes for packets by means of hosts capable of handling them (as defined by the security policy) if the normal routing is prohibited by the interconnection security policy.

Session Management Services

The session management services equate to the secure association service of the framework and provide for:

privilege attribute transport
This service propagates the privilege attributes of each principal between the platform domains as part of the creation of an association (session). The principals in the formation of the association are processes (or more correctly the threads of execution within the processes) and the privilege attributes are the process attributes of those communicating processes.

authentication
Authentication of platform domains and principals (service domains or applications) is indirectly supported by the assertions on protection of the physical network and the security responsibilities of the interconnecting platform domains and their management (see Section C.2.3 on page 167).

access control
Authorisation for formation of the association has to be confirmed by the interconnection policies enforced by both communicating platform domains. The authorisation is determined on the basis of the security attributes of the association request, the security attributes of the peer processes, the security attributes of the host platforms, and the security policies enforced by each platform domain.

audit
The session management services ensure that the network audit events defined by the security policy are detected and recorded.

Message Security Attribute Services

The message security attribute services comprise:

message security attributes
The message security attribute service ensures that a set of security attributes are associated with every IP packet. These security attributes are derived, by default, from the process attributes of the originating process.

modify message default security attributes
An appropriately authorised process may modify the default security attributes that are applied to messages it originates.

set and retrieve per message security attributes
A suitably authorised process may set the security attributes of each message it originates on a per message basis.

A suitably authorised process may retrieve the security attributes of any message it receives.

per message access control
The platform domain enforces access control on a per message basis and only delivers messages to processes that are authorised to receive those messages as determined by the security attributes of the message and the process attributes of the destination process.

Inter-domain Services

The TSIX(RE) specification supports interdomain services by means of a set of databases and administrative interfaces for their management.

These databases provide for the configuration of the interconnection security policies between the platform domains comprising the network and mappings of security attributes representations within each platform domain. Where an individual platform domain does not support a particular aspect of policy and the associated security attribute, the inter-domain service may support the assignment of a default security attribute value to messages originating from that domain.

A further inter-domain service is provided to applications to support the exchange of security attributes in an implementation independent format.

C.2.2 TSIX Interfaces

The TSIX(RE) specification is based on extensions to communication service access point interfaces such as sockets and XTI. They support both security unaware and security aware callers.

Security Unaware Callers

Security unaware callers use the standard, unmodified communication service interfaces. The TSIX implementation is required to derive all necessary information from the process attributes of the communicating peer processes.

Security Aware Callers

Appropriately authorised security aware callers can:

- Configure a communication service access point to handle multiple security contexts and make the message security attribute services available. That is the caller is authorised to enforce some aspects of the security policy and the TSIX service passes responsibility for this to the caller.
- Set default message security attributes to be applied to all messages subsequently sent.
- Set the security attributes of messages originated on a per message basis.
- Retrieve the security attributes of messages retrieved.
- Build messages containing security attributes in a system-independent manner and extract security attributes from such a message in a local system representation.

Management Interfaces

Management interfaces are provided to support the administration of the TSIX inter-domain databases. These interfaces map to the security framework services for administering interconnection security policies including permitted associations and inter-domain mappings.

C.2.3 Assertions on Environment

The TSIX(RE) specification makes the following assertions on its environment, that is all platform domains on the network:

- All platform domains that are to interact support the same protocol suite (IP).
- The implementation labels exported packets with the correct IP address.
- The implementation imports only those IP packets addressed to the host platform domain.
- Access to communication system end-points (service access points) is restricted to authorised processes. The access control policy is assumed to include:
  - — restricting access to end-points associated with reserved Internet addresses (reserved for specific service domains)
  - — restricting access to Internet protocols other than UDP and TCP (for example, raw IP and ICMP, EGP, and so on); this is an assertion that the TSIX services cannot be bypassed.
  - — restricting access to Interfaces or network protocols below the transport layer. This is again an assertion that the TSIX services cannot be bypassed.
  - — preventing a process from accessing an end-point that it did not create unless properly delegated to do so by the creator; that means the creator of an end-point is able to assign and control access rights to that end-point.
- Each platform domain, and supporting service domains, is required:
  - — properly to authenticate users interfacing to the system by means of that platform domain and assign appropriate privilege attributes to that user
  - — to assign processes' security attributes that properly represent the principal's identity and privilege attributes, whether the principal is a user or service domain.
  - — implement the TSIX services such that the correct process attributes are exchanged during TSIX session establishment.
- The TSIX implementation within a platform domain is required to apply a specified set of default security attributes to messages originated outside the domain.
- The physical network is protected from external threats.

Table C-1 Example Mapping of Standards and Implementations to Framework

	ISO POSIX-1	**UNIX**	**P1003.1e**	**TSIX(RE)**	**DCE 1.0**	**DCE1.1**
Domain	Platform	Platform Additional platform security services: print, mail	Platform	Platforms	DCE cells DCE apps.	Same as DCE 1.0
Elements	Processes Files	Processes Files, IPCs Infrastructure service objects (for example, mail, print objects)	Processes Files	Processes End points	DCE principals DCE objects DFS files	Same as DCE 1.0

	ISO POSIX-1	**UNIX**	**P1003.1e**	**TSIX(RE)**	**DCE 1.0**	**DCE1.1**
Security attributes	uids, gids	uids, gids	uids, gids, labels, capabilities, audit id, audit mask	uids, gids, labels, capabilities, audit id, audit mask	principal uuid, group uuids	Same as DCE 1.0 plus Extended Registry Attributes (ERAs)
Security attributes bind/retrieve	setuid, getuid, setgid, getgid	setuid, getuid, setgid, getgid	setuid, getuid, setgid, getgid, mac_set_proc, mac_get_proc, cap_set_proc, cap_get_proc	Set defaults for end-point m6def_attr; read attributes from messages; m6peek, m6last_attr	set_login_setup_ identity, sec_login_ validate_identity, sec_login_set_ context, sec_login_get_ current_context, sec_login_ newgroups, sec_login_get_ groups, sec_login_get_ pwent, sec_login_inquire_ net_info, rpc_binding_set_ auth_info, rpc_binding_inq_ auth_info, rpc_if_register_ auth_info, rpc_binding_inq_ auth_client	Same as DCE 1.0 plus: set_login_set_ extended_attrs, sec_login_become _initiator, sec_login_become _delegate, rpc_binding_inq_ auth_caller, sec_login_cred_ get_initiator, sec_login_cred_ get_delegate, sec_cred_get_*
Cryptographic and key management	N/A	crypt	N/A	N/A	N/A	N/A
Authentication	N/A	getpwent	N/A	N/A; based on mutual trust of all platforms and their services	Kerberos V5, sec_login_validate _identity, sec_login_valid_ and_cert, sec_key_mgt_*	Same as DCE 1.0 plus pre-authentication options
Authorisation	Implicit platform services only; restricted ACL; chmod, chown		ACLs acl_set_file acl_get_file Labels mac_set_file mac_get_file capabilities	Delivery of message from end point to process controlled	DCE ACLs sec_acl_* rpc_mgmt_set_ authorization_fn	Same as DCE 1.0
Security Audit	N/A	utmp etc.	Platform services aud_write aud_read	Network events; location of audit trail	No	Audit of security events, dce_aud_*

	ISO POSIX-1	**UNIX**	**P1003.1e**	**TSIX(RE)**	**DCE 1.0**	**DCE1.1**
Security attribute acquisition and verification	Not needed	N/A	Not needed	N/A	Implicit in authenticated RPC	Same as DCE 1.0 plus explicit in GSS-API
Privilege attribute service	getpwent	getpwent	getpwent (uid, gid only)	N/A	Implicit in DCE PS; explicit in all services listed under DCE 1.0 in row entitled Security Attributes bind/retrieve	Implicit in DCE PS; explicit in all services listed under DCE 1.0 in row entitled Security Attributes bind/retrieve
Interdomain service	N/A	NFS, NTS	N/A	Implicit Domains database Mappings database	Establish intercell trust with rgy_edit cell command, passwd_import, passwd_export, passwd_override, sec_rgy_acct_*	Same as DCE 1.0 plus transitive trust with hierarchical cells, non-DCE security domain integration with ERAs, sec_rgy_attr_*
Certification authority	N/A	N/A	N/A	N/A	N/A	N/A
Key distribution service	N/A	N/A	N/A	N/A	N/A	N/A
Trusted third party services	N/A	N/A	N/A	N/A	N/A	N/A
Sponsor service	N/A	login and server part of telnet and ftp services	N/A	N/A	DCE login	Same as DCE 1.0
Sign-on	N/A	login and server part of telnet and ftp services	N/A	N/A	DCE login	Same as DCE 1.0
Secure association service	N/A	N/A	N/A	Supports secure exchange of security attributes	Implicit in authenticated RPC	Same as DCE 1.0 plus explicit in GSS-API

Appendix D

Security Interfaces to Standards

This appendix outlines an example set of criteria on which to assess:

- the relative merits of standardising an interface to a particular security service

  This requires the identification of the relative benefits and costs to both vendors and users (including integrators) of both standardising and not standardising the proposed interfaces.

- the relative merits of particular candidate API specifications as a base implementation for a standard

  This requires consideration of the current status of the specification both in terms of acceptability within standards bodies and in terms of implemented and deployed products together with any interoperability and IPR or other issues associated with the specification or implementations thereof.

To illustrate the example, an API for the secure association service is taken as a proposed interface for standardisation and the GSS-API as a potential candidate for such an API.

D.1 Standardisation Area

Description and Scope	API for integration of secure associations into distributed processing services.
Consequences of Support	
Benefit for Vendor	Potentially wider market.
Benefit for User	Portability of distributed processing services (TP, ftp, telnet, ORB) across different security mechanisms. Avoids need for distributed processing services to implement trusted security mechanisms. Allows interoperability (only if the supported key distribution protocols are compatible).
Cost for Vendor	Implementation cost small (compared with existing mechanisms). If confidentiality supported then products must support exportable algorithms.
Cost for User	Existing distributed processing services must change to use API. If confidentiality supported then use of product may be subject to legislative controls in some countries.
Consequences of Non-support	
Benefit for Vendor	Save small implementation effort.
Benefit for User	None.
Cost for Vendor	Potentially smaller market.
Cost for User	Lock in of distributed processing services to specific security mechanism.
Commercial Value to Vendor	Open systems conformance marketing message. The providers of distributed processing services are the (first) customer. Mass market is only indirectly accessed (by the inclusion of licensed technology).

D.2 Candidate for Base API

The candidate is GSS-API as published in the **GSS-API Base** specification.

Status		
Documents		Proposed Standard IETF RFC. X/Open Preliminary Specification; this is to be developed as a branded component with conformance tests. Imported into ISO SC21/WG8 Secure Association Management.
Products		Q494 — to be implemented by OSF/DCE 1.1 vendors over DCE 1.1. 1995 — to be implemented in Microsoft Windows NT over Kerberos V5. now — has been implemented in IBM NetSP over KryptoKnight. now — has been implemented in OCSG/CyberSafe Kerberos over Kerberos V5. Q195 — to be implemented by SESAME vendors over SESAME.
Issues		
Interoperability		Interoperability depends upon the compatibility of the security mechanisms over which the the **GSS-API Base** specification is implemented. Currently, the implementations only interoperate with identical implementations with qualified interoperability between Kerberos V5 based implementations and DCE 1.1 or SESAME based implementations.
IPR	Interface	GSS-API is not encumbered with any IPR issues.
	Security Mechanism	The specification of a security mechanism may be published or proprietary and subject to government or trade-secret constraints.
	Implementation	An implementation of a security mechanism may be licensable.

Appendix E

Example Architectural Models

This appendix provides simplified examples of the approach to be taken to include security into the specification of primary service APIs within different classes of service.

The recommended approach is to:

- construct an architectural model of the functional service
- describe the security environment in which it executes
- assign responsibility for security policy enforcement to the individual components of the architectural model
- identify which components interface to additional security services.

The examples used are based on the X/Open Transaction Processing Model and an X.400 service. These examples are illustrative only, are not complete and leave many details and questions unanswered. It the responsibility of the appropriate working groups to develop these models with the help of security experts.

E.1 Security Model of Transaction Processing

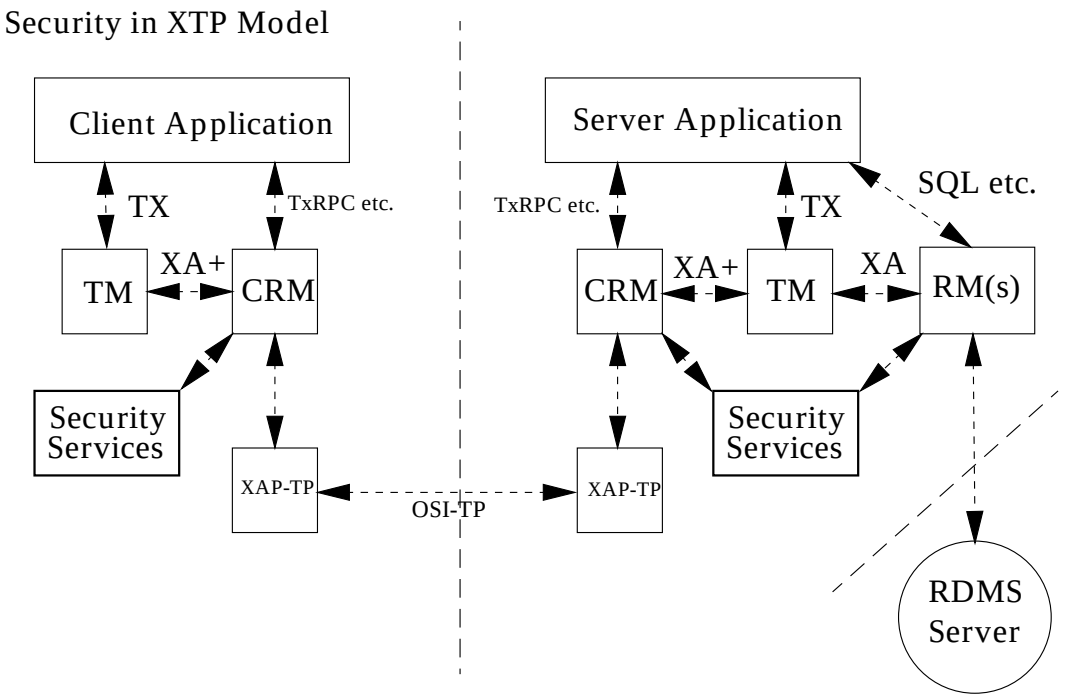

Figure E-1 Transaction Processing Security Model

The XTP model comprises three manager components that are assumed to execute in the same process as the application:

Transaction Manager (TM)
This is the coordinating component responsible for handling the concept of a transaction.

Communication Resource Manager (CRM)
This component is responsible for the management of the communication medium, including the creation of associations.

Resource Managers (RMs)
Resource managers are responsible for the management of the data storage services, for example databases.

The X/Open model defines three sets of interfaces:

- TX between the application and the TM
- XA between the TM and the RMs
- XA+ between the TM and the CRM.

Client Security Environment

The client is assumed to operate within a security context that has been established by the User Sign-on service and bound to the client application process.

Server Security Environment

The server is assigned its own identity and may possess credentials with which to authenticate that identity. The server operates with its own security context for some operations, for example, access to its own platform resources, but must be capable of associating a client's privilege attributes with operations performed on behalf of the client.

Client Application

The client application does not handle the security context itself. It is security unaware.

Server Application

The server application is assumed to be capable of servicing multiple clients. It must therefore be able to associate the appropriate privilege attributes of the client principal with each operation performed.

Communication Resource Manager

The CRMs of both the client and the server are responsible for the formation of associations and therefore the establishment of the security context for each association. To support this responsibility the CRMs invoke the secure association service.

The client application CRM is responsible for retrieving a handle to the security context bound to the client process and passing this to the secure association service to generate the exportable security context (security token).

The server application CRM is responsible for passing the imported security context to the secure association service and receiving a handle to that context. It is then responsible for binding that security context to the thread of execution within the server application representing that client.

Transaction Manager

The TM manages the transaction context of operations on behalf of the applications. The security context is transparent to the TM; however, the TM may need to alter the security context for some operations such as recovery.

Resource Manager

The RM is responsible for invoking the services of the data stores supporting the server application. It is responsible for retrieving the privilege attributes of the client principal and using these as the basis of authorisation for the use of the resources.

The resource manager may need to invoke the security services in order to delegate security credentials to remote components of the RM.

E.2 Example of Model for Security in Messaging Applications

E.2.1 Functional Model of X.400

X.400 is an example of a store and forward messaging system and a simplified model of the X.400 messaging system is illustrated in Figure E-2.

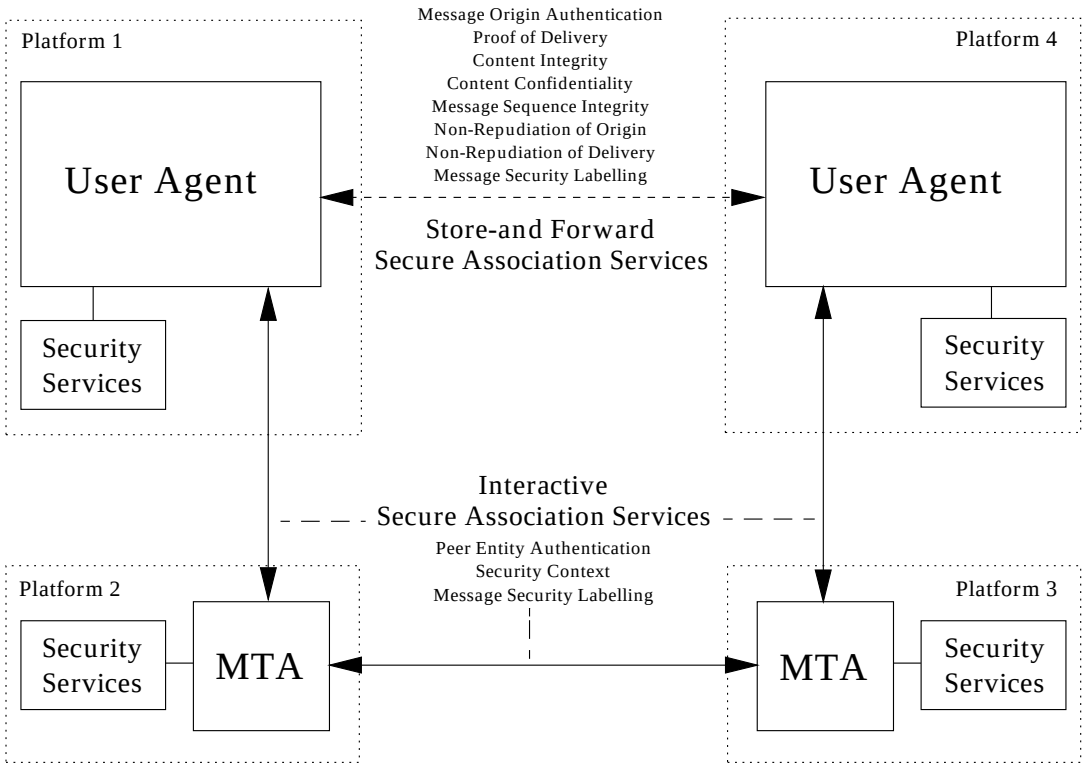

Figure E-2 X400 Simplified Model

Users interact with the system via User Agent (UA) services. A user (initiator) sends a message to another user (recipient) by preparing a message. A message comprises a body party, the ''text'' of the message, together with appropriate addressing information to identify the location of the recipient.

The User Agent submits the message to its local X.400 Message Transfer Agent (MTA). The MTA routes the message by means of other MTAs to its ultimate destination, a User Agent representing the recipient.

The message is transferred by a series of interactive associations between peer processes, UA to MTA, MTA to MTA and MTA to UA. This series of interactive associations supports a virtual association between the message initiator and the message recipient. This virtual association is a store and forward based association.

E.2.2 Security Model of X.400

The X.400 specification identifies a set of security services related to the store and forward based association between the Initiating User Agent and the Recipient User Agent. These services are end-to-end services and comprise:

- message origin authentication
- proof of delivery
- content integrity
- content confidentiality
- message sequence integrity
- non-repudiation of origin
- non-repudiation of delivery
- message security labelling.

These end to end security services rely upon cryptographic mechanisms to bind a set of security information, representing the security context of the association, with the message.

The X.400 specification identifies a set of security services related to the interactive associations between peer agents; that is, between a UA and an MTA, between a MTA and a MTA, between a MTA and a UA. These node to node based security services are:

- peer entity authentication
- security context establishment and message authorisation
- message security labelling.

These node to node security services are based upon a negotiation of the security context between the two communicating peers by way of a security protocol.

E.2.3 Security Implications on APIs

Any interfaces supporting interactions between a User and a User Agent must support user interfaces to the associated security services. For example, the ability to request a particular Quality of Protection or Non-repudiation Services. Associated management services must also support the management of security-related information, for example, the directory service must support the storage and retrieval of a entity's public key.

APIs providing an interface to an MTA (for example as might be incorporated into an application providing User Agent functions) must support the invocation of the Secure Association Services on association creation.

E.3 Security Model of DCE

DCE provides a distributed computing framework in which applications can communicate securely with one another across a network using a Remote Procedure Call (RPC) paradigm. The basic elements of the DCE security model are the application client and server. The server provides the client with the service or action it requests. The server can use Access Control Lists (ACLs) to allow or deny requests from specific clients.

The DCE Security Service has three core elements:

- the Registry, which maintains the database of principals and their associated long-term keys
- the Key Distribution Service, which manages the authentication of principals and the distribution of the credential certificates (tickets) for specific servers
- the Privilege Service, which delivers Privilege Attribute Certificates (PACs) containing the attributes used by servers to control access based on their ACLs

These three elements are provided by the DCE Security Server. The DCE Security Service is partitioned in basic units called cells. A cell corresponds to one instance of the DCE Security Server (and its replicas, if configured). The DCE Security Service offers mechanisms to support both intra-cell and inter-cell operations.

Before a principal can participate in a secure DCE operation, it must have an identity registered with the Security Service, and it must have established its identity with DCE. End-users establish their identities through the DCE login facility. They provide their password (proof of identity) to the local DCE Security Service agent, which returns login credentials. Clients typically obtain their *targeted* credentials by providing the login credentials they inherited from the end-user login to the Security Service. Servers, which cannot provide a password, establish their identities by the means of the Key Management Facility.

Figure E-3 on page 181 illustrates the interaction between the components of the DCE Security Service, the clients and the RPC. Note that the Login Facility, the Key Management Facility, the Auditing Facility, and the various other administration tools (Registry, ACL) are not shown for simplicity:

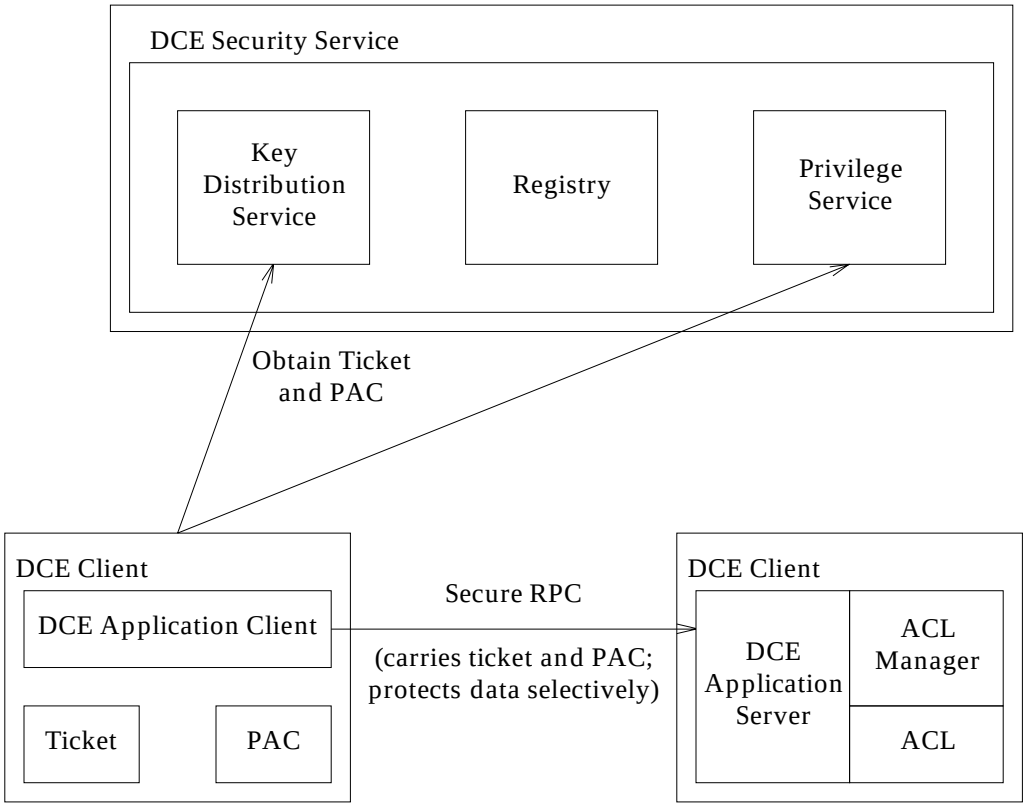

Figure E-3 Interaction between DCE, Clients and RPC

The Secure RPC uses the targeted credentials established by the client in its first request to the server. The server's ACL Manager uses them to verify access by matching the PAC with the ACL for the specific request. The RPC communication is protected for authenticity, integrity and confidentiality as agreed by the client and the server. Note that the mechanisms used ensure a two-way authentication, where both the client and the server can rely on the identity of the other party.

The DCE Security Service is also accessible as a mechanism of the Generic Security Service API (GSS-API). Application developers wishing to use the DCE authentication, integrity or confidentiality mechanisms can do so by using either the DCE RPC or other communication paradigms in conjunction with the GSS-API.

Other DCE services, such as Time and Naming, use the DCE Security Service to ensure the DCE framework offers a consistent level of security.

Appendix F

Security in API Specification

This appendix provides a template for an element covering security requirements in X/Open specifications.

In the template explanatory text is marked with shading in the margin.

Appendix n

Security

This appendix defines the security aspects of this specification. The issues are discussed as relevant threats and concerns, and the impact on other specifications. An overview and details of the security specification are provided.

n.1 Security Issues

n.1.1 Threats

With reference to Section 3.5 on page 23 and Appendix A, identify the security threats due to the introduction of new resources or capability from this specification. Do not include the text from Section 3.5 on page 23 and Appendix A.

n.1.2 Basic Security Policy Requirements

With reference to Section 1.1.2 on page 2 identify the relevant basic security policy requirements and put them in order of priority:

- integrity
- availability
- confidentiality
- accountability.

None is a possibility.

n.1.3 Impact on Other Specifications

Identify the potential security impact on other specifications, such as what assumptions are being made about the characteristics of provider services.

n.2 Overview of Security Solution

n.2.1 Security Goals

Identify the security goals. What specific security is required of implementors conforming to this specification, for example, what resources are protected against which threats?

n.2.2 Security Framework

Using the examples in Appendix E, apply the framework to create a security architecture for the subject of the specification. Specify the level of awareness of interfaces with respect to specific services.

n.2.3 Security Functionality and Services

List the applicable services from Chapter 5 and Chapter 6.

List any additional security functionality needed.

n.2.4 Standards

Identify applicable security standards that may be relevant to this specification.

n.2.5 Emerging Standards

Identify applicable security efforts currently underway that may be relevant to this specification now or in the future.

n.3 Security Specification

With reference to the items listed in Appendix C, specify in detail the security functionality required to provide protection of the features of this specification.

n.3.1 Domain

n.3.2 Elements

n.3.3 Security Attributes

n.3.4 Security Attributes Bind and Retrieve

n.3.5 Cryptographic and Key Management

n.3.6 Authentication

n.3.7 Authorisation

n.3.8 Security Audit

n.3.9 Security Attribute Acquisition and Verification

n.3.10 Privilege Attribute Service

n.3.11 Interdomain Service

n.3.12 Certification Authority

n.3.13 Key Distribution

n.3.14 Trusted Third Party Services

n.3.15 Sponsor Service

n.3.16 User Sign-on

n.3.17 Secure Association Service

Glossary

access control
The prevention of unauthorised use of a resource including the prevention of use of a resource in an unauthorised manner (see ISO / IEC 7498-2).

access control certificate
ADI in the form of a security certificate (see ISO / IEC 10181-3).

access control decision function
(ADF) — a specialised function that makes access control decisions by applying access control policy rules to a requested action, ACI (of initiators, targets, actions, or that retained from prior actions), and the context in which the request is made (see ISO / IEC 10181-3).

access control decision information
(ADI) —) the portion (possibly all) of the ACI made available to the ADF in making a particular access control decision (see ISO / IEC 10181-3).

access control enforcement function
(AEF) — a specialised function that is part of the access path between an initiator and a target on each access that enforces the decisions made by the ADF (see ISO / IEC 10181-3).

access control information
(ACI) — any information used for access control purposes, including contextual information (see ISO / IEC 10181-3).

access control list
A list of entities, together with their access rights which are authorised to have access to a resource (see ISO / IEC 7498-2).

access control policy
The set of rules that define the conditions under which an access may take place (see ISO / IEC 10181-3).

accountability
The property that ensures that the actions of an entity may be traced to that entity (see ISO / IEC 7498-2).

ACI
Access control information.

ACL
Access control list.

action
The operations and operands that form part of an attempted access (see ISO / IEC 10181-3).

action ADI
Action decision information associated with the action (see ISO / IEC 10181-3).

active threat
The threat of a deliberate unauthorised change to the state of the system (see Appendix A on page 145).

ADF
Access control decision function.

ADI
Access control decision information.

administrative security information
Persistent information associated with entities; it is conceptually stored in the Security Management Information Base. Examples are:

- security attributes associated with users and set up on user account installation, which is used to configure the user's identity and privileges within the system
- information configuring a secure interaction policy between one entity and another entity, which is used as the basis for the establishment of operational associations between those two entities.

AEF
Access control enforcement function.

alarm collector function
A function that collects the security alarm messages, translates them into security alarm records, and writes them to the security alarm log (see ISO / IEC 10181-7).

alarm examiner function
A function that interfaces with a security alarm administrator (see ISO / IEC 10181-3).

API
Application Programming Interface.

The interface between the application software and the application platform, across which all services are provided.

The application programming interface is primarily in support of application portability, but system and application interoperability are also supported by a communication API (see POSIX.0).

assertion
Explicit statement in a system security policy that security measures in one security domain constitute an adequate basis for security measures (or lack of them) in another (see CESG Memorandum No 1).

association-security-state
The collection of information that is relevant to the control of communications security for a particular application-association (see ISO / IEC 10745).

audit
See Security Audit (see ISO / IEC 7498-2).

audit authority
The manager responsible for defining those aspects of a security policy applicable to maintaining a security audit (see ISO / IEC 10181-7).

audit event detector function
A function that detects the occurrence of security-relevant events. This function is normally an inherent part of the functionality implementing the event (see ISO / IEC 10181-7).

audit recorder function
A function that records the security-relevant messages in a security audit trail (see ISO / IEC 10181-7).

audit trail
See Security Audit Trail (see ISO / IEC 7498-2).

audit trail analyser function
A function that checks a security audit trail in order to produce, if appropriate, security alarm messages (see ISO / IEC 10181-7).

audit trail archiver function
A function that archives a part of the security audit trail (see ISO / IEC 10181-7).

audit trail collector function
A function that collects individual audit trail records into a security audit trail (see ISO / IEC 10181-7).

audit trail examiner function
A function that builds security reports out of one or more security audit trails (see ISO / IEC 10181-7).

audit trail provider function
A function that provides security audit trails according to some criteria (see ISO / IEC 10181-7).

authenticated identity
An identity of a principal that has been assured through authentication (see ISO / IEC 10181-2).

authentication
Verify claimed identity; see data origin authentication, and peer entity authentication (see ISO / IEC 7498-2).

authentication certificate
Authentication information in the form of a security certificate which may be used to assure the identity of an entity guaranteed by an authentication authority (see ISO / IEC 10181-2).

authentication exchange
A sequence of one or more transfers of exchange authentication information (AI) for the purposes of performing an authentication (see ISO / IEC 10181-2).

authentication information (AI)
Information used to establish the validity of a claimed identity (see ISO / IEC 7498-2).

authentication initiator
The entity which starts an authentication exchange (see ISO / IEC 10181-2).

authentication method
Method for demonstrating knowledge of a secret. The quality of the authentication method, its strength is determined by the cryptographic basis of the key distribution service on which it is based. A symmetric key based method, in which both entities share common authentication information, is considered to be a weaker method than an asymmetric key based method, in which not all the authentication information is shared by both entities.

authorisation
The granting of rights, which includes the granting of access based on access rights (see ISO / IEC 7498-2).

authorisation policy
A set of rules, part of an access control policy, by which access by security subjects to security objects is granted or denied. An authorisation policy may be defined in terms of access control lists, capabilities or attributes assigned to security subjects, security objects or both (see ECMA TR / 46).

availability
The property of being accessible and usable upon demand by an authorised entity (see ISO / IEC 7498-2).

capability
A token used as an identifier for a resource such that possession of the token confers access rights for the resource (see ISO / IEC 7498-2).

ciphertext
Data produced through the use of encipherment. The semantic content of the resulting data is not available (see ISO / IEC 7498-2).

Note: Ciphertext may itself be input to encipherment, such that super-enciphered output is produced.

claim authentication information
(Claim AI) — information used by a claimant to generate exchange AI needed to authenticate a principal (see ISO / IEC 10181-2).

claimant
An entity which is or represents a principal for the purposes of authentication. A claimant includes the functions necessary for engaging in authentication exchanges on behalf of a principal (see ISO / IEC 10181-2).

clear text
Intelligible data, the semantic content of which is available (see ISO / IEC 7498-2).

client-server
These operations occur between a pair of communicating independent peer processes. The peer process initiating a service request is termed the client. The peer process responding to a service request is termed the server. A process may act as both client and server in the context of a set of transactions.

confidentiality
The property that information is not made available or disclosed to unauthorised individuals, entities, or processes (see ISO / IEC 7498-2).

contextual information
Information derived from the context in which an access is made (for example, time of day) (see ISO / IEC 10181-3).

corporate security policy
The set of laws, rules and practices that regulate how assets including sensitive information are managed, protected and distributed within a user organisation (see ITSEC).

countermeasure
The deployment of a set of security services to protect against a security threat.

credentials
Data that is transferred to establish the claimed identity of an entity (see ISO / IEC 7498-2).

cryptanalysis
The analysis of a cryptographic system and its inputs and outputs to derive confidential variables and / or sensitive data including clear text (see ISO / IEC 7498-2).

cryptographic algorithm
A method of performing a cryptographic transformation (see cryptography) on a data unit. Cryptographic algorithms may be based on symmetric key methods (the same key is used for both encipher and decipher transformations) or on asymmetric keys (different keys are used for encipher and decipher transformations).

cryptographic checkvalue
Information that is derived by performing a cryptographic transformation (see cryptography) on a data unit (see ISO / IEC 7498-2).

Note: The derivation of the checkvalue may be performed in one or more steps and is a result of a mathematical function of the key and data unit. It is usually used to check the integrity of a data unit.

cryptography
The discipline that embodies principles, means, and the methods for the transformation of data in order to hide its information content, prevent its undetected modification and/or prevent its unauthorised use (see ISO / IEC 7498-2).

Note: The choice of cryptography mechanism determines the methods used in encipherment and decipherment. An attack on a cryptographic principle, means or methods is cryptanalysis.

data integrity
The property that data has not been altered or destroyed in an unauthorised manner (see ISO / IEC 7498-2).

data origin authentication
The corroboration that the entity responsible for the creation of a set of data is the one claimed.

decipherment
The reversal of a corresponding reversible encipherment (see ISO / IEC 7498-2).

decryption
See decipherment (see ISO / IEC 7498-2).

denial of service
The unauthorised prevention of authorised access to resources or the delaying of time-critical operations (see ISO / IEC 7498-2).

digital fingerprint
A characteristic of a data item, such as a cryptographic checkvalue or the result of performing a one-way hash function on the data, that is sufficiently peculiar to the data item that it is computationally infeasible to find another data item that possesses the same characteristics (see ISO / IEC 10181-1).

digital signature
Data appended to, or a cryptographic transformation (see cryptography) of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery for example, by the recipient (see ISO / IEC 7498-2).

discretionary access control
A discretionary authorisation scheme is one under which any principal using the domain services may be authorised to assign or modify ACI such that he may modify the authorisations of other principals under the scheme. A typical example is an ACL scheme which is often referred to as Discretionary Access Control (DAC).

distinguishing identifier
Data that unambiguously distinguishes an entity in the authentication process. Such an identifier shall be unambiguous at least within a security domain (see ISO / IEC 10181-2).

distributed application
A set of information processing resources distributed over one or more open systems which provides a well-defined set of functionality to (human) users, to assist a given (office) task (see ECMA TR / 46).

encapsulated subsystem
A collection of procedures and data objects that is protected in a domain of its own so that the internal structure of a data object is accessible only to the procedures of the encapsulated subsystem and that those procedures may be called only at designated domain entry points. Encapsulated subsystem, protected subsystem and protected mechanisms of the TCB are terms that may be used interchangeably (see Federal Criteria).

encipherment
The cryptographic transformation of data (see cryptography) to produce ciphertext (see ISO / IEC 7498-2).

Note: Encipherment may be irreversible, in which case the corresponding decipherment process cannot feasibly be performed. Such encipherment may be called a one-way-function or cryptochecksum.

encryption
See encipherment (see ISO / IEC 7498-2).

end-to-end encipherment
Encipherment of data within or at the source end system, with the corresponding decipherment occurring only within or at the destination end system (see ISO / IEC 7498-2).

exchange authentication information
(Exchange AI) — information exchanged between a claimant and a verifier during the process of authenticating a principal (see ISO / IEC 10181-2).

identification
The assignment of a name by which an entity can be referenced. The entity may be high level (such as a user) or low level (such as a process or communication channel.

identity-based security policy
A security policy based on the identities or attributes of users, a group of users, or entities acting on behalf of the users and the resources or targets being accessed (see ISO / IEC 7498-2).

initiator
An entity (for example, human user or computer based entity) that attempts to access other entities (see ISO / IEC 10181-3).

initiator access control decision information
(Initiator ADI) — ADI associated with the initiator (see ISO / IEC 10181-3).

initiator access control information
(Initiator ACI) — access control information relating to the initiator (see ISO / IEC 10181-3).

integrity
See Data Integrity (see ISO / IEC 7498-2).

key
A sequence of symbols that controls the operations of encipherment and decipherment (see ISO / IEC 7498-2).

key management
The generation, storage, distribution, deletion, archiving and application of keys in accordance with a security policy (see ISO / IEC 7498-2).

masquerade
The unauthorised pretence by an entity to be a different entity (see ISO / IEC 7498-2).

messaging application
An application based on a store and forward paradigm; it requires an appropriate security context to be bound with the message itself.

non-discretionary access control
A non-discretionary authorisation scheme is one under which only the recognised security authority of the security domain may assign or modify the ACI for the authorisation scheme such that the authorisations of principals under the scheme are modified.

off-line authentication certificate
A particular form of authentication information binding an entity to a cryptographic key, certified by a trusted authority, which may be used for authentication without directly interacting with the authority (see ISO / IEC 10181-2).

on-line authentication certificate
A particular form of authentication information, certified by a trusted authority, which may be used for authentication following direct interaction with the authority (see ISO / IEC 10181-2).

operational security information
Transient information related to a single operation or set of operations within the context of an operational association, for example, a user session. Operational security information represents the current security context of the operations and may be passed as parameters to the operational primitives or retrieved from the operations environment as defaults.

organisational security policy
Set of laws, rules, and practices that regulates how an organisation manages, protects, and distributes sensitive information (see Federal Criteria).

password
Confidential authentication information, usually composed of a string of characters (see ISO / IEC 7498-2).

peer-entity authentication
The corroboration that a peer entity in an association is the one claimed (see ISO / IEC 7498-2).

physical security
The measures used to provide physical protection of resources against deliberate and accidental threats (see ISO / IEC 7498-2).

platform domain
A security domain encompassing the operating system, the entities and operations it supports and its security policy.

policy
See security policy (see ISO / IEC 7498-2).

primary service
An independent category of service such as operating system services, communication services and data management services. Each primary service provides a discrete set of functionality. Each primary service inherently includes generic qualities such as usability, manageability and security.

Security services are therefore not primary services but are invoked as part of the provision of primary services by the primary service provider.

principal
An entity whose identity can be authenticated (see ISO / IEC 10181-2).

privacy
The right of individuals to control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

Note: because this term relates to the right of individuals, it cannot be very precise and its use should be avoided except as a motivation for requiring security (see ISO / IEC 7498-2).

private key
A key used in an asymmetric algorithm. Possession of this key is restricted, usually to only one entity (see ISO / IEC 10181-1).

public key
The key, used in an asymmetric algorithm, that is publicly available (see ISO / IEC 10181-1).

quality of protection
A label that implies methods of security protection under a security policy. This normally includes a combination of integrity and confidentiality requirements and is typically implemented in a communications environment by a combination of cryptographic mechanisms.

repudiation
Denial by one of the entities involved in a communication of having participated in all or part of the communication (see ISO / IEC 7498-2).

rule-based security policy
A security policy based on global rules imposed for all users. These rules usually rely on a comparison of the sensitivity of the resources being accessed and the possession of corresponding attributes of users, a group of users, or entities acting on behalf of users (see ISO / IEC 7498-2).

seal
A cryptographic checkvalue that supports integrity but does not protect against forgery by the recipient (that is, it does not support non-repudiation). When a seal is associated with a data element, that data element is *sealed* (see ISO / IEC 10181-1).

secondary discretionary disclosure
An example of the misuse of access rights. It occurs when a principal authorised to access some information copies that information and authorises access to the copy by a second principal who is not authorised to access the original information.

secret key
In a symmetric cryptographic algorithm the key shared between two entities (see ISO / IEC 10181-1).

secure association
An instance of secure communication (using communication in the broad sense of space and/or time) which makes use of a secure context.

secure context
The existence of the necessary information for the correct operation of the security mechanisms at the appropriate place and time.

secure interaction policy
The common aspects of the security policies in effect at each of the communicating application processes (see CESG Memorandum No 1).

security architecture
A high level description of the structure of a system, with security functions assigned to components within this structure (see CESG Memorandum No 1).

security attribute
A security attribute is a piece of security information which is associated with an entity.

security audit
An independent review and examination of system records and operations in order to test for adequacy of system controls, to ensure compliance with established policy and operational procedures, to detect breaches in security and to recommend any indicated changes in control, policy and procedures (see ISO/IEC 7498-2).

security audit message
A message generated following the occurrence of an auditable security-related event (see ISO/IEC 10181-7).

security audit record
A single record in a security audit trail corresponding to a single security-related event (see ISO/IEC 10181-7).

security audit trail
Data collected and potentially used to facilitate a security audit (see ISO/IEC 7498-2).

security auditor
An individual or a process allowed to have access to the security audit trail and to build audit reports (see ISO/IEC 10181-7).

security aware
The caller of an API that is aware of the security functionality and parameters which may be provided by an API.

security certificate
A set of security-relevant data from an issuing security authority that is protected by integrity and data origin authentication, and includes an indication of a time period of validity (see ISO/IEC 10181-1).

Note: All certificates are deemed to be security certificates (see the relevant definitions in ISO/IEC 7498-2) adopted in order to avoid terminology conflicts with ISO/IEC 9594-8 (that is the directory authentication standard).

security domain
A set of elements, a security policy, a security authority and a set of security-relevant operations in which the set of elements are subject to the security policy, administered by the security authority, for the specified operations (see ISO/IEC 10181-1).

security event manager
An individual or process allowed to specify and manage the events which may generate a security message and to establish the action or actions to be taken for each security message type (see ISO / IEC 10181-7).

security label
The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource (see ISO / IEC 7498-2).

Note: The marking may be explicit or implicit.

security policy
The set of criteria for the provision of security services (see also identity-based and rule-based security policy).

security service
A service which may be invoked directly or indirectly by functions within a system that ensures adequate security of the system or of data transfers between components of the system or with other systems.

security state
State information that is held in an open system and which is required for the provision of security services.

security token
A set of security-relevant data that is protected by integrity and data origin authentication from a source that is not considered a security authority (see ISO / IEC 10181-1).

security unaware
The caller of an API that is unaware of the security functionality and parameters which may be provided by an API.

sensitivity
The characteristic of a resource that implies its value or importance, and may include its vulnerability (see ISO / IEC 7498-2).

separation
The concept of keeping information of different security classes apart in a system (see CESG Memorandum No 1).

Note: Separation may be implemented by temporal, physical, logical or cryptographic techniques.

service domain
A security domain encompassing an application, the entities and operations it supports and its security policy.

signature
See digital signature (see ISO / IEC 7498-2).

strength of mechanism
An aspect of the assessment of the effectiveness of a security mechanism, namely the ability of the security mechanism to withstand direct attack against deficiencies in its underlying algorithms, principles and properties (see ITSEC).

system security function
A capability of an open system to perform security-related processing (see CESG Memorandum No 1).

target
An entity to which access may be attempted (see ISO / IEC 10181-3).

target ADI
ADI associated with the target (see ISO / IEC 10181-3).

target ACI
Access control information relating to the target (see ISO / IEC 10181-3).

threat
A potential violation of security (see ISO / IEC 7498-2).
An action or event that might prejudice security (see ITSEC).

traffic analysis
The inference of information from observation of traffic flows (presence, absence, amount, direction and frequency) (see ISO / IEC 7498-2).

traffic flow confidentiality
A confidentiality service to protect against traffic analysis (see ISO / IEC 7498-2).

traffic padding
The generation of spurious instances of communication, spurious data units or spurious data within data units (see ISO / IEC 7498-2).

trap door
A hidden software or hardware mechanism that permits system protection mechanisms to be circumvented. It is activated in some non-apparent manner (for example, special ''random'' key sequence at a terminal) (see TCSEC).

trojan horse
Computer program containing an apparent or actual useful function that contains additional (hidden) functions that allow unauthorised collection, falsification or destruction of data (see Federal Criteria).

trust
A relationship between two elements, a set of operations and a security policy in which element X trusts element Y if and only if X has confidence that Y behaves in a well defined way (with respect to the operations) that does not violate the given security policy (see ISO / IEC 10181-1).

trusted computing base (TCB)
The totality of protection mechanisms within an IT system, including hardware, firmware, software and data, the combination of which is responsible for enforcing the security policy.

trusted functionality
That which is perceived to be correct with respect to some criteria, for example, as established by a security policy (see ISO / IEC 7498-2).

trusted path
Mechanism by which a person using a terminal can communicate directly with the TCB (see Federal Criteria).

Note: Trusted path can only be activated by the person or the TCB and cannot be imitated by untrusted software.

trusted third party
A security authority or its agent, trusted by other entities with respect to security-related operations (see ISO / IEC 10181-1).

verification AI
Information used by a verifier to verify an identity claimed through exchange AI (see ISO / IEC 10181-2).

verifier
An entity which is or represents the entity requiring an authenticated identity. A verifier includes the functions necessary for engaging in authentication exchanges (see ISO / IEC 10181-2).

virus
Self replicating, malicious program segment that attaches itself to an application or other executable system component and leaves no external signs of its presence (see Federal Criteria).

vulnerability
Weakness in an information system or components (for example, system security procedures, hardware design, internal controls) that could be exploited to produce an information-related misfortune (see Federal Criteria).

Index