X/Open Technical Study

Security in Interworking Specifications

X/Open Company Ltd.

X/Open Technical Study

Security in Interworking Specifications

ISBN: 1-85912-051-2
X/Open Document Number: E403

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 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 an X/Open Technical Study. Security in the context of this document is defined as the need to protect information and resources from damage and unauthorised use. It is a vital consideration for the owners and users of information processing systems. It is of particular concern for systems that use communications media, because of the increased access to systems and information that these media provide.

In order to promote the greater security of Open Systems, each X/Open interface specification addresses security considerations. This Technical Study complements the the referenced **X/Open Distributed Security Framework** (see referenced documents). While the Framework addresses general security issues, this study concentrates on those security problems that are specific to interworking. After discussing the security requirements common to all X/Open interworking definitions, it gives security considerations specific to the XTI, DCE RPC and XDS specifications.

Intended Audience

This Technical Study does not contain a complete explanation of security issues. It aims to be self-standing, in that it introduces all the concepts that are required in order to understand it, but those concepts are a subset of those presented in the the referenced **X/Open Distributed Security Framework**. In particular, it does not explore issues associated with different forms of security policy, with security domains, or with the concepts of trust and assurance. Readers concerned with these issues should consult the referenced **X/Open Distributed Security Framework**.

Structure

- Chapter 1 introduces the basic terminology and scope of this technical study
- Chapter 2 describes the threats to interworking security and the countermeasures that can be taken against them
- Chapter 3 describes how the countermeasures, and control of them, can be implemented
- Chapter 4 describes the security information that is required to support them

- Chapter 5 describes the underlying security services that they use
- Chapter 6 describes how far security can be achieved through the X/Open interworking APIs, as currently specified
- Chapter 7 describes protocols that support secure interworking but that are not currently referenced by X/Open interworking specifications
- Chapter 8 describes the conclusions that can be drawn.

These chapters are followed by three appendices which give examples of security considerations in existing X/Open Interworking specifications.

Referenced Documents

The following documents are referenced in this Technical Study:

(PC) NFS
: Protocols for X/Open Interworking: (PC)NFS, Developers' Specification, XO/DEV/90/030, ISBN: 1-872630-00-6, The X/Open Company Ltd., 1990

ACSE
: ISO/IEC 8650: 1988, Information Technology - Open Systems Interconnection - Protocol Specification for the Association Control Service Element

ACSE Authentication
: ISO/IEC 8650 Amd 1: 1990, Authentication during association establishment

BSFT
: Byte Stream File Transfer (BSFT), CAE Specification, XO/CAE/91/400, ISBN: 1-872630-27-8, The X/Open Company Ltd., 1991

CLI
: Data Management: SQL Call Level Interface (CLI), Snapshot, S203, ISBN: 1-872630-63-4, The X/Open Company Ltd., 1992

CMIP
: ISO/IEC 9596-1: 1991, Common Management Information Service Protocol

COTS-over-TCP
: ISO Transport Service on top of the TCP, version 3 - RFC 1006

DCE Directory
: X/Open DCE: Directory Services, Preliminary Specification, to be published.

DCE RPC
: DCE: Remote Procedure Call, Preliminary Specification, P312, ISBN: 1-872630-95-2, The X/Open Company Ltd., 1993

DCE Security
: AES/Distributed Computing - Security, Revision A - Preliminary Version, The Open Software Foundation, 1994

DNS
: Domain Names - Implementation and Specification, RFC 1035

FTP
: File Transfer Protocol (FTP) - RFC 959

GSSAPI C Bindings RFC
: Generic Security Service API: C Bindings - RFC-1509

GSSAPI RFC
: Generic Security Service Application Program Interface - RFC-1508

GULS
: ISO/IEC DIS 11586, Information Technology - Open Systems Interconnection - Generic Upper Layers Security (Parts 1-4)

IP
: Internet Protocol - RFC 791

IPC for SMB
: IPC Mechanisms for SMB, CAE Specification, XO/CAE/91/500, ISBN: 1-872630-28-6, The X/Open Company Ltd., 1991

Internet Mail Privacy
: Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures - RFC 1421 (see also RFCs 1422, 1423 and 1424)

Minimal OSI
: ISO/IEC pDISP 11188, Information Technology - Open Systems Interconnection - Common Upper Layer Requirements, Part 3: Minimal OSI upper layer facilities

NETBIOS
: Mappings of NetBIOS to OSI and IPS transport protocols are described in the referenced **IPC for SMB** CAE Specification.

NLSP
: ISO/IEC 11577, Information Technology - Open Systems Interconnection - Network Layer Security Protocol

OSI CLTS
: ISO 8072 Add. 1, Information Technology - Open Systems Interconnection - Transport Service Definition Addendum 1: Connectionless mode

OSI COTS
: ISO 8072, Information Technology - Open Systems Interconnection - Transport Service Definition

OSI Security Architecture
: ISO 7498-2, Information Technology - Open Systems Interconnection - Basic Reference Model - Part2: Security Architecture

OSI TP
: ISO/IEC 10026, Information Technology - Open Systems Interconnection - Distributed Transaction Processing

OSI Transport Security Association
: ISO/IEC 10736/DAM 1, Information Technology - Open Systems Interconnection - Transport Layer Security Protocol Amendment 1: Security association establishment

OSI Upper Layer Security Model
: ISO/IEC CD 10745, Information Technology - Open Systems Interconnection - Upper Layer Security Model

SMB
: Protocols for X/Open PC Interworking: SMB, version 2, CAE Specification, C209, The X/Open Company Ltd., 1992

SMTP
: Simple Mail Transfer Protocol - RFC 821

SNMP
: The Simple Network Management Protocol, RFC 1157

SNMP v2
: Introduction to Version 2 of the Internet-Standard Network Management Framework, RFC 1441

SNMP v2 Security
: Security Protocols for Version 2 of the Simple Network Management Protocol, RFC 1446

SQL
Structured Query Language (SQL), CAE Specification, C201, ISBN: 1-872630-58-8, The X/Open Company Ltd., 1992

Secure Systems Procurement
Defining and Buying Secure Open Systems, X/Open Guide, G206, ISBN: 1-872630-61-8, The X/Open Company Ltd., 1992

TCP
Transmission Control Protocol, RFC 793

TLSP
ISO/IEC 10736, Information Technology - Open Systems Interconnection - Transport Layer Security Protocol

Telnet
Telnet Protocol Specification - RFC 854

UDP
User Datagram Protocol, RFC 768

X.25
CCITT Recommendation X.25: Interface between DCE and DTE for Terminals Operating in the Packet Mode and Connected to Public Data Networks by Dedicated Circuit

X.400
The CCITT X.400 series recommendations are concerned with Message Handling Systems. They are technically aligned with ISO/IEC 10021, Information processing systems - Text communication - Message oriented text interchange system

X.500
The CCITT X.500 series recommendations are concerned with the Directory. They are technically aligned with ISO/IEC 9594, Information Technology - Open Systems Interconnection - The Directory

XAP
ACSE/Presentation Services API (XAP), CAE Specification, C303, ISBN: 1-872630-91-X, The X/Open Company Ltd., 1993

XDCS
Distributed Computing Services (XDCS) Framework, X/Open Guide, G212, ISBN: 1-872630-64-2, The X/Open Company Ltd., 1992

XDS
API to Directory Services (XDS), CAE Specification, XO/CAE/91/090, ISBN: 1-872630-18-9, The X/Open Company Ltd., 1991

XFTAM
FTAM High-Level API (XFTAM), Preliminary Specification, P206, ISBN: 1-872630-60-X, The X/Open Company Ltd., 1992

XMHS
API to Electronic Mail (X.400) CAE Specification, XO/CAE/91/100, ISBN: 1-872630-19-7, The X/Open Company Ltd. in conjunction with the X.400 API Association, 1991

XMP
Systems Management: Management Protocols API (XMP), Preliminary Specification, P170, ISBN: 1-872630-32-4, The X/Open Company Ltd., 1992

XMS
Message Store API (XMS) Preliminary Specification, XO / PRELIM / 91 / 060, ISBN: 1-872630-26-X, The X / Open Company Ltd. in conjunction with the X.400 API Association, 1992

XNFS
Protocols for X / Open Interworking: XNFS, CAE Specification, XO / CAE / 91 / 030, ISBN: 1-872630-10-3, The X / Open Company Ltd., 1991

XOM
OSI-Abstract-Data Manipulation API (XOM) CAE Specification, XO / CAE / 91 / 080, ISBN: 1-872630-17-0, The X / Open Company Ltd. in conjunction with the X.400 API Association, 1991

XTI
X / Open Transport Interface (XTI), CAE Specification, XO / CAE / 91 / 600, ISBN: 1-872630-29-4, The X / Open Company Ltd., 1992

X / Open GSS-API
Generic Security Service Application Programming Interface, Preliminary Specification, The X / Open Company Ltd., 1993

X / Open GSS-API Extensions
Generic Security Service Extensions, Snapshot, The X / Open Company Ltd., 1993

X / Open Distributed Security Framework
X / Open Distributed Security Framework, X / Open Guide, Document No. G410, ISBN 1-85912-071-7, The X / Open Company Ltd., October 1994.

X Protocol
X / Open Window Management: X Window System Protocol, CAE Specification, XO / CAE / 91 / 050, ISBN: 1-872630-13-8, The X / Open Company Ltd., 1991

Xlib
X / Open Window Management: Xlib - C Language Binding, CAE Specification, XO / CAE / 91 / 040, ISBN: 1-872630-11-1, The X / Open Company Ltd., 1991

Chapter 1

Introduction

Security - the need to protect information and resources from damage and unauthorised use - is a vital consideration for the owners and users of information processing systems. It is of particular concern for systems that use communications media, because of the increased access to systems and information that these media provide.

In order to promote the greater security of Open Systems, each X/Open interface specification addresses security considerations. This Technical Study complements the the referenced **X/Open Distributed Security Framework** (see referenced documents). While the Framework addresses general security issues, this study concentrates on those security problems that are specific to interworking. After discussing the security requirements common to all X/Open interworking definitions, it gives security considerations specific to the XTI, DCE RPC and XDS specifications.

This Technical Study does not contain a complete explanation of security issues. It aims to be self-sufficient, in that it introduces all the concepts that are required in order to understand it, but those concepts are a subset of those presented in the referenced **X/Open Distributed Security Framework**, and the study does not explore them in such depth as is covered in the referenced **X/Open Distributed Security Framework**. different forms of security policy, with security domains, or with the concepts of trust and assurance. The reader concerned with these issues should consult the referenced **X/Open Distributed Security Framework**.

1.1 Interface Specifications and Components

X/Open publishes *interface specifications*, but brands products on the basis of their conformance to *component definitions*. For example, an implementation of the referenced **XTI** CAE Specification is branded as a Transport Services (XTI) component.

A *component* of the X/Open CAE is a functional unit whose interfaces are defined in specifications listed in the X/Open Portability Guide (the XPG). These specifications may be published by X/Open or by other bodies These specifications may be published by X/Open or by other bodies (for example, they may be international standards published by ISO).

A component as defined in the XPG can have one or more interfaces of the following kinds:

- human/computer interfaces used for interaction with human users
- portability interfaces used for interaction with other components and applications in the same computer (these interfaces are also referred to as *programming interfaces* or, when they are used by application programs, as *application program interfaces* or APIs)
- communications interfaces used for interaction across networks with other components and applications in a distributed system, and with other systems.

1.2 Communications Components

This Technical Study is concerned with components that have communications interfaces (in the sense of the XPG); these will be referred to as *communications components*. It is not restricted to those components that are currently defined in the XPG. It considers all functional units that have interfaces specified by X/Open and that also have communications interfaces. Any such unit could potentially be an XPG component, and may be referred to as a *component* in this Technical Study.

Human users interacting with a component via human/computer interfaces, and other components or applications interacting with a component via portability interfaces, will be referred to as *users* of the component. Other components and applications in a distributed system, or other systems, that interact with a component via a communications interface will be referred to as *communications partners*.

A component definition may include by reference several interface specifications. Typically, the definition of a communications component refers to one or more communications interface specifications as well as to a programming interface specification. For example, the XPG4 Transport Services (XTI) component definition requires the component to support the ISO Transport communications interface, the TCP/IP communications interface, or the UDP/IP communications interface, as well as the XTI programming interface.

1.3 Threats and Countermeasures

Threats that are specific to communications are more likely to be associated with communications interfaces than with programming interfaces. However, most of the interface specifications defined by X/Open are for programming interfaces. The communications interface specifications referred to in the X/Open component definitions are typically defined by other bodies, such as ISO and CCITT.

In defining a countermeasure to a security threat, all of the interfaces of a component must be considered. For example, there may be little point in adding a *confidentiality requested* argument to a function in a programming interface specification unless the corresponding communications interface specification allows encryption.

Threats relating to communications arise even for components that do not have programming interfaces or user interfaces. The XPG4 (PC)NFS Server is an example of such a component.

1.4 Security Domains

In the referenced X/Open Security Framework, the term *security domain* is used to describe a part of an operational system that forms a unit for security purposes, and to which a set of countermeasures is applied against threats. For each security domain, there is a *security policy*[1] that determines what countermeasures are applied, and how they are applied. For each security domain, there is also a *security authority* that administers the security policy.

An operational system typically contains a number of security domains. They may be nested (for example, the whole system could form a domain and the disk subsystem could be a subdomain of it). They may overlap. For example, a database management system and application could form one domain, and the disk subsystem could form another. These domains overlap because they both contain the part of the disk store that is used by the database management system. However, neither is a subdomain of the other.

In an operational system, a communications component typically forms a part of several overlapping security domains. The component must be able to support part of the countermeasures required by the security policies of the domains that include it.

A security domain that includes a communications component typically also includes the communications networks used by the component. The component supports part of the countermeasures that the security policies require to be implemented against threats arising through use of the networks.

1. This use of the term *security policy* refers to the security policy of a user enterprise. It must be distinguished from the *security implementation policy* (for a component) that is described in Chapter 3.

   An analysis and discussion of security policies is contained in the referenced **X/Open Distributed Security Framework**. It is the *interdomain* aspect of security policy that is of most relevance for this Technical Study.

1.5 Overall Framework

In defining countermeasures to security threats, it is desirable to consider all the components in a system together, as well as considering each one singly. This avoids *chinks in the armour* where threats that affect several components are not completely countered by their combined security measures. It also avoids developing different mechanisms that do the same job, but for different components.

Security issues should therefore be addressed for components, rather than for interfaces, and should be addressed within the context of an overall system architecture or framework.

The Security Framework which the X/Open security working group is developing provides the general context in which security aspects of every individual component should be addressed. For components associated with communications or networking, there is also the X/Open Distributed Computing Services (XDCS) Framework, described in the referenced **XDCS** Framework specification.

The XDCS Framework is not concerned with components, but with services. The services of the XDCS Framework do not correspond exactly to the components defined in XPG4, because:

- there are some services (for example, Time Services) that the XDCS Framework identifies, but for which no XPG components are yet defined
- there are some cases where a *service* identified by the XDCS Framework should more properly be considered as a related group of services, each of which could be provided by a separate XPG component (for example, the XDCS *Messaging Services* include an X.400 gateway service that could be provided by the XPG4 X.400 Gateway component and an X.400 message access service that could be provided by the XPG4 X.400 Message Access component).

The XDCS Framework does cover some aspects of security, but does not cover all the security needs of all components. It includes Security Services for authentication (as in Kerberos), authorisation (using Access Control Lists), and protection (using cryptography). In the context of XDCS, these services are designed for use in conjunction with Remote Procedure Call (RPC) services. Whether these Security Services are sufficient in the wider context of the XDCS Framework as a whole, and how far they can be used in conjunction with communications components other than those providing RPC Services, are among the questions that this Technical Study addresses.

Chapter 2

Threats and Countermeasures

A *threat* is a possibility of damage to or unauthorised use of information or resources. This chapter addresses threats related to interworking. More precisely, it is concerned with threats that arise because systems include communications components. These threats are analysed, and the countermeasures to them are discussed.

Communications components may have human/computer interfaces and programming interfaces as well as communications interfaces. There are threats that arise from human/computer interaction and from interaction across programming interfaces. These threats are outside the scope of this Technical Study. There are also threats that are associated with particular communications components but not with communications components in general. These threats also are not covered here. Instead, they are addressed in the security requirements for individual interface specifications.

2.1 Threats

The types of harm that can be done are listed in the referenced **X/Open Distributed Security Framework**. They are:

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

The ways in which these types of harm can be caused through communications components are discussed below. They include most of the threats listed in the referenced **OSI Security Architecture** standard. (Some of the threats listed there - *insider attacks*, *trapdoor* and *Trojan horse* - are outside the scope of this Technical Study, for reasons discussed above).

2.1.1 Unauthorised Modification

Information may be modified as a result of a masquerade in which an unauthorised communications partner impersonates an authorised one. This may be achieved by establishing a new connection or association, or by inserting data into the data stream of an existing connection or association.

Information may be modified as a result of tampering with a communications medium, for example through the introduction of unauthorised resources or the misuse of management facilities. The modification could be achieved by destroying, corrupting, substituting, replaying or resequencing transmitted information.

Information may be modified as a result of a security breach in a remote system.

2.1.2 Unauthorised Disclosure

Information may be disclosed as a result of a masquerade.

Information may be disclosed as a result of monitoring or tampering with a communications medium.

Information may be disclosed as a result of traffic analysis on communications networks. Generally, this is a more serious threat for military applications than for commercial, industrial or administrative applications.

Information may be disclosed as a result of a security breach in a communications partner.

Note that disclosure of information may result in opportunities for masquerade attack by replaying disclosed information (possibly after modifying it) at a later time.

2.1.3 Unauthorised Use of Resources

Services provided to communications partners may be used by unauthorised users as a result of masquerade.

Services provided to communications partners may be used by an unauthorised user presenting himself as though he were authorised.

2.1.4 Denial of Service

The services of a communications partner may be made unavailable by damage to communications networks.

The services of a communications partner may be made unavailable by congestion in communications networks.

Note that congestion in or damage to a communications network can be produced, for example, by the introduction of unauthorised resources or the misuse of management facilities. Congestion can also be produced by saturation of resources resulting from excessive use. The saturated resource may be in a remote system, in a communications network, or in the communications infrastructure in the local system.

2.1.5 Repudiation

The originator of information sent across a network may subsequently deny having sent it in the form in which it was received (or may deny having sent it at all). Such information may, for example, have financial or legal significance.

The recipient of information may deny having received it in the form in which it was sent.

2.2 Countermeasures

The referenced **X/Open Security Framework** describes a set of generic security mechanisms that provide for the security of a security domain by ensuring that:

- inputs to the domain can only be made, and outputs from the domain can only be received, through the identified interfaces of the domain (*domain segregation*)
- the sources of inputs and destinations of outputs are authentic, the inputs are received from them correctly, and the outputs are received by them correctly (*information verification*)
- inputs and outputs can only be made when the principals on whose behalf they are made are authorised to request them (*service mediation*)
- the services of the domain are available when needed (*resilience and recovery*)
- a record is kept of activity relevant to security (*activity recording*).

A communications component fulfills its responsibilities for implementing the generic security mechanisms that ensure the security of the security domains that include it by implementing some of the security mechanisms identified in the referenced **OSI Security Architecture** standard. These mechanisms apply to communications that do not use OSI protocols just as well as they apply to communications that do use them[2]. The relevant mechanisms[3] are discussed in the subsections below.

2.2.1 Authentication Exchange

This is the exchange of security information in order to verify the claimed identity of a communications partner. It is used to counter masquerade attempts made by establishing new connections or associations. It is part of the *information verification* generic countermeasure.

Typically, the exchanged information includes either passwords or encrypted tokens. (These are small information items, possibly including timestamps. Successful decryption of a token implies that it was encrypted using a particular key that a masquerader would not have known.) Transmission of passwords across communications networks is not very secure; they can be disclosed through masquerade or through network monitoring. Encrypted tokens are therefore to be preferred.

There are two approaches to authentication exchange in distributed systems:

Bilateral Agreement
Each pair of communications partners establishes a means by which information supplied by one can be authenticated by the other (for example, a shared secret encryption key).

Third Party Authentication
A third party authenticates information supplied by the sender (using the bilateral approach) and provides evidence of this to the recipient. The recipient verifies (again, using the bilateral approach) that the evidence has come from the third party.

More generally, there can be a chain of third parties. The first authenticates the information supplied by the sender and passes evidence of this to the next one in the chain. Each

2. However, other aspects of the OSI Security Architecture are less applicable outside the context of OSI.

3. *Trusted Functionality* is not discussed. This mechanism is used to ensure that the hardware or software that provides, or provides access to, other security mechanisms is trustworthy. The means by which this may be done are outside the scope of this Technical Study. They include security-oriented design and implementation reviews and formal proofs of correctness.

member of the chain authenticates the evidence supplied by the preceding member and passes evidence to the next member. The last member of the chain passes evidence to the recipient of the information.

Two different third party authentication schemes have been defined for distributed open systems:

- the certification authority scheme of the X.500 directory service
- the Kerberos ticket granting scheme used in the Distributed Computing Environment (DCE) defined by the Open Software Foundation (OSF) and specified by X/Open as part of the XDCS.

When two partners communicate, either partner can be threatened by a masquerade of the other. Authentication of the identities of communications partners should therefore be mutual.

More generally, when more than two partners communicate (for example, using broadcast protocols), any partner can be threatened by a masquerade of another (although not necessarily by a masquerade of any other). Mutual authentication (although not necessarily of all pairs of communicators) is again required.

A form of authentication that is commonly used over dial-up links is *dial-back*. In this scheme, the called party breaks down the initial network connection, and sets up a new one, using a network address (in most cases, this is a telephone number) that it has previously stored as the calling party's address.

2.2.2 Access Control

Access control is the ability to grant or deny access to a resource. In the context of interworking security, the form of access control considered is that which is applied to an attempt to gain access across a communications network. It is part of the *service mediation* generic countermeasure.

Access is granted or denied on the basis of:

- the identity of the entity requesting access (the *principal* - this is often the human user who invoked the application)
- other information about the principal (such as claimed membership of particular user groups)
- the type of access requested.

Access control is used to prevent unauthorised users obtaining services by presenting themselves as though they were authorised.

2.2.3 Data Integrity

Data integrity protection is achieved by performing an integrity check on an item of data and attaching the result to the data. The recipient can re-create the result by performing the integrity check on the data received, and hence can determine whether it has been modified.

The data integrity security mechanism is part of the *domain segregation* and *information verification* generic countermeasures.

An integrity check can cover an individual message or a series of messages. In the latter case, it guards against deletion and replay of messages.

It must not be possible to forge the result of the integrity check. A result can be protected from forgery by combining a digital signature with it. This is the only form of integrity check that is considered in this Technical Study.

Data integrity can be used to counter attempts to modify information by tampering with communications media.

2.2.4 Encipherment

Encipherment, also called *encryption*, is performed by transforming information in a way that can be reversed only by its intended recipient. This is normally done using an algorithm that is publicly known but that has a parameter known as its *key*, such that the key required to reverse the transformation (that is, to *decypher* or *decrypt* the information) is not publicly known.

Encypherment is part of the *domain segregation* generic countermeasure.

There are a number of encryption techniques. An exhaustive survey is beyond the scope of this Technical Study. Two general classes can be distinguished, however:

- "**Secret Key Encryption**,"
  in which the information is encrypted using a key that is only known to the parties concerned, and
- "**Public Key Encryption**,"
  in which the information is encrypted using a key that is publicly known (the *public key*) and decrypted using a key (the *private key*) known only to the receiver.

Most public key encryption schemes also have the property that information encrypted using the private key can be decrypted using the public key.

For secret key encryption, there must be a *key distribution service* that informs recipients of the keys used by senders to encrypt data. This service must be secure.

Public key encryption does not require a secure key distribution service, since the keys that must be distributed can be disclosed without harm. However, to protect against some forms of masquerade, a *key certification service* is required to ensure that the keys received across a network are genuine.

Encryption can be applied to:

- security information (such as integrity check results)
- communications protocol information (for example, network addresses)
- user data.

Since it is not generally possible to modify an encrypted message without knowing the key, encryption includes integrity checking of individual messages. However, it does not prevent replay or deletion of messages in a message stream.

Encryption can be used to counter:

- attempts to disclose data by monitoring or tampering with communications media
- some attempts to modify information by tampering with communications media (it does not protect against deletion or replay of messages)
- traffic analysis (when the encryption is applied to communications protocol information).

Encryption can be used as part of other security mechanisms, including authentication, integrity protection and digital signature.

2.2.5 Digital Signature

Typically, a *digital signature* consists of an encrypted token (see Section 2.2.1 on page 8.) It is added to a piece of information so that its origin can be verified. This is part of the *information verification* generic countermeasure.

If it is possible to modify the signed information without this being detected, then the signature is worthless. Digital signature should therefore always be used in conjunction with integrity protection or with encryption (which provides integrity protection).

Used together with data integrity protection, digital signature provides protection against attempts to masquerade by inserting data into the data stream of an existing connection or association.

If the verification can be performed by a third party (neither the sender nor the receiver), then a digital signature provides protection against repudiation.

2.2.6 Notarisation

Notarisation is the certification by a third party that a piece of information (such as a digital signature) is genuine. It is used in conjunction with digital signatures to protect against repudiation. It is part of the *information verification* generic countermeasure.

2.2.7 Security Labelling

Security labelling is achieved by adding to transmitted information further information about its security attributes. It is used as part of most of the other security mechanisms described in this section (for example, an encrypted token can be considered as a form of security label). However, there is an important security attribute that does not form a part of any of the other security mechanisms. This is the degree of *sensitivity* of the information (for example: *secret*, *confidential* or *unclassified*).

Security labelling for sensitivity (*sensitivity labelling*) can provide some protection against a security breach in a communications partner, since the partner can apply more stringent protection measures to more sensitive information. Sensitivity labelling can also be used by one form of routing control (see below).

2.2.8 Routing Control

Traffic can be made to take particular routes, for example:

- through networks that are physically secure
- that are thought not to be subject to attack
- that will not result in a particularly noticeable traffic pattern.

Routing control can be carried out in two ways:

- in some circumstances (for example, when the network protocols support source routing), the communications components can determine the routes to be followed
- network nodes can route information on the basis of sensitivity labels transmitted with the information.

Routing control can provide protection against traffic analysis, monitoring of networks, or tampering with networks. It is part of the *domain segregation* generic countermeasure.

2.2.9 Traffic Padding

This is the generation of spurious communications or spurious data within communications. It provides protection against traffic analysis. It is part of the *domain segregation* generic countermeasure.

Other security mechanisms against this threat - routing control and encryption of protocol information - are not always possible because of network limitations (for example, most networks can not handle encrypted address information). Traffic padding can still be used in such cases.

2.2.10 Event Detection

This is the detection of events relevant to security. It is part of the *activity recording* and *resilience and recovery* generic countermeasures. Relevant events can include attempted security violations, normal protocol events and API events.

Event detection may lead to taking action, for example by forcing disconnection, generating alarms, changing encryption keys, or logging the event as part of an audit trail. This does not prevent breaches of security. It may prevent an attacker from capitalising on them by catching him red-handed.

2.2.11 Audit Trail

An audit trail is created by recording all the significant events that take place in a transaction. While this does not in itself prevent breaches of security, it does make it easier to detect the people responsible, and therefore can act as a deterrent. It also helps a systems manager, when there has been a breach of security, to assess the damage that has been done and prevent it from being repeated. It is part of the *activity recording* generic countermeasure.

2.2.12 Security Recovery

This is the taking of appropriate action to recover from actual or attempted security violations detected by functions such as event handling or systems management. It is part of the *resilience and recovery* generic countermeasure. Actions similar to those described for Event Detection (in which violations are detected by the communications function) can be taken.

2.3 Adequacy of Security Mechanisms

There are specific security mechanisms that cover all of the identified threats, except for:

- threats of modification or disclosure of information in a system that arise from security breaches in communications partners (this threat is partly, but not completely, covered by sensitivity labelling).

  There is little that can be done in one system to counter security breaches in another. Applications should be designed such that information is not distributed unnecessarily. Organisations should ensure that their systems are secure, and should require other organisations with which they communicate to ensure that their systems are secure also. They should create a situation in which there is mutual trust that a joint policy for interworking security[4] will be followed.

- threats of denial of service due to damage to networks or network congestion.

  Protection against damage and congestion can only be obtained by following good network design and management procedures. For example, network design can provide alternate routing.

In addition, there are the general event detection, audit trail and security recovery mechanisms that provide some protection agains threats of most types.

Work in distributed systems security over the last few years has not disclosed any significant new threats or identified any new security mechanisms. There is no guarantee that other threats do not exist. However, the security mechanisms discussed in this chapter, with the limitations stated in this section, constitute the best currently known protection.

4. Refer to the referenced **X/Open Distributed Security Framework** for a discussion of security policy. The type of security policy referred to here is an interdomain security policy.

Chapter 3

Implementation of Security Mechanisms

This chapter discusses some aspects of the practical implementation of security mechanisms in a system that includes X/Open communications components.

3.1 Security Implementation Policy

For each component, there will be particular requirements for implementing security mechanisms. These will largely arise from the nature of the applications that might use the component. There will also be particular considerations affecting how the security mechanisms can be implemented. The statement of what security mechanisms should be implemented in a component, and how they should be implemented, in the light of those requirements and considerations, forms its *security implementation policy*. For each X/Open interworking specification, the common security implementation policy requirements for components supporting the specification are discussed in its security appendix.

3.2 Placement of Security Functionality

The first question that arises is, where is the security functionality implemented? Are security programs part of the application, part of the communications programs, or part of some other software?

The extent to which a communications component should apply security mechanisms depends on the needs of the applications that will use the component. If a large number of applications require protection against a particular threat then it makes sense for the component to implement the security mechanisms, or to invoke other components that do so. Conversely, if a threat is not likely to be serious for many applications then it makes sense for the applications to apply the security mechanisms, in the rare cases where they are needed.

In practice, there is a limit on the extent to which communications components can apply security mechanisms. Many of the security mechanisms require information to be conveyed between communications partners (see Section 4.1 on page 22 or to the network infrastructure (see Section 4.2 on page 23). For those security mechanisms to be implemented by the communications components, this information must be conveyed explicitly by the communications protocol. (This means, not as user data. Security information can always be conveyed as user data, but such data is handled by applications, not by the communications components).

Not all protocols are capable of conveying the necessary information. The extent to which protocols referenced by X/Open interworking specifications can convey security information is summarised in the tables in Chapter 6.

The security mechanisms include functionality that is not communications-related. (For example, encryption of the information is not a communications activity, although the placing of encrypted information in a protocol data unit is a communications activity). This functionality can be considered as a set of *underlying security services*. The underlying security services relevant to interworking are described in Chapter 5.

Currently, the X/Open specifications and component definitions do not refer to the functionality of the security services. It can either be incorporated within communications components or be provided by other components that are invoked by communications components.

3.3 Control of Countermeasures

For those security mechanisms that are applied by communications components, there are a number of ways of determining *when* they are applied.

Feature Options

Support for a security mechanism can be optional, so that some implementors provide it while others do not.

Optional features are described in X/Open interworking specifications, and support for them must be described by implementors when they complete the associated conformance statement questionnaires.

Configuration Parameters

Parameters set when a system is installed or re-configured can determine which security mechanisms are applied, and under what circumstances. For example, they might enable encryption to be switched on or off for all traffic that uses a particular port.

No X/Open interworking specifications currently specify this method of control for security mechanisms.

Environment Variables

A communications component might read environment variables set by the user in order to determine whether to apply a security mechanism. For example, an environment variable might determine whether outgoing electronic mail messages are to be encrypted, and a user could turn it on for a whole session or just for an individual message.

No X/Open interworking specifications currently specify this method of control for security mechanisms.

System Management

System management facilities[5] can be used to turn security mechanisms on or off at run time. For example, they might enable encryption to be switched on or off for all traffic with a particular communications partner.

Any system management facilities used to control security mechanisms must themselves be secure.

No system management facilities for controlling security mechanisms are currently specified by X/Open.

Control by Applications

Applications can use APIs to determine when security mechanisms will be applied. They can use the communications APIs or separate security APIs. For example, an application might use an API to turn encryption on or off for a particular transaction.

5. Refer to the referenced **X/Open Distributed Security Framework** for a deeper analysis of system management aspects, and their relation to security policy.

Each user enterprise has different needs, which are reflected in its *security policy*[6] The security policy of a user enterprise could call for a security mechanism to be used:

- always
- for particular users
- for particular communications partners
- for particular types of transaction.

Within X/Open-compliant systems, it is rare for security mechanisms to be controlled by configuration parameters or environment variables. This probably reflects the security-policy needs of most users. The most important distinction is between control of security mechanisms by applications and control of security mechanisms through systems management.

Applications that control security mechanisms are called *security-aware*. This concept is discussed in more depth in the referenced **X/Open Distributed Security Framework**, to which the reader is referred for further details.

Security-aware applications can provide the most flexibility in implementing security policies of user enterprises. However, a user enterprise may wish to use an application that is not security aware, or it may not wish to trust a security-aware application.

Existing applications are typically not security-aware. This situation is expected to continue. The preferred means of controlling security mechanisms is likely to be through systems management facilities. It is a requirement that system administrators should be able to control the application of security mechanisms for security unaware applications.

Whether control is applied by applications or by systems management components, a programmatic interface can be used as the means of applying this control. This programmatic interface could be a part of the communications API or could be a separate API.

Some of the communications APIs specified by X/Open enable applications to determine when security mechanisms are applied. These are listed in the tables in Chapter 6 on page 31. No additional security APIs to communications components are currently defined by X/Open.

Where control of a security mechanism that could be applied by a communications component is not provided by the existing communications API, it would be possible to extend the API to include control of the security mechanism, or to define a separate API for this purpose.

Having a separate API would allow the development of a common security mechanisms control API that could be used by all interworking components. This would have the advantages that applications programmers would not have to learn a new set of functions for each API, that less specification effort would be required, that existing API specifications would not all have to be changed, and that there could be common functionality which would allow implementors of multiple components to save effort.

Having a separate API would also enable user enterprises to deny its use to their programming staff. This could be an important requirement of some security policies.

6. The term *security policy* (of a user enterprise) used here should not be confused with *security implementation policy* (for a component) used elsewhere in this chapter.

For these reasons, X/Open believes that if APIs to control security mechanisms in communications components are to be specified, then the possibility should be explored of defining a common security control API.

Chapter 4

Security Information

This chapter discusses the information that is required to support the security mechanisms identified in Section 2.2 on page 8. It falls into three classes:

- information that is transmitted between communications partners using communications protocols
- information that is transmitted using communications protocols but is used by the network infrastructure rather than by communications partners
- information that is not transmitted.

4.1 Information Transmitted between Communications Partners

4.1.1 Identification Information

Identification information is used to identify users, services and systems. Identification information can take any of the following forms.

Names
Human-readable character strings, for example, Internet host names.

Numeric Identities
Numbers that identify entities, for example *uids* in X/Open-compliant systems. They usually apply within the context of a single system.

Formal Identities
Bit strings that conform to some formal representation scheme, for example ASN.1 Object Identifiers (OIDs) or DCE Universal Unique Identifiers (UUIDs).

Directory Names
Names that can be used to find information about the entities in a directory, for example X.500 directory names.

4.1.2 Authentication Information

This is information that is used to verify the origin of a piece of information. It may consist of:

- a password, that is, an item of information known only to the authentication service and the originator of the information
- information that can only have been created by the originator of the information, such as an encrypted token.

As a protection against replay, an encrypted token may include a sequence number or a timestamp.

4.1.3 Attribute Information

This is information about an entity that can be used for the purpose of controlling access by that entity to a resource. It may, for example, include claims of group membership. DCE Privilege Attribute Certificates (PACs) contain attribute information.

4.1.4 Access Type Information

This information specifies a type of access that is requested to a resource controlled by an application. Access types are application-dependent. For example, a filestore manipulation application might have access types of *read*, *append* and *write*.

4.1.5 Integrity Information

This is information that is associated with data for integrity protection purposes. Such information may include:

- algorithm identifiers for integrity checks
- session identifiers and message sequence numbers (to protect against modification by deletion or replay of individual messages)
- integrity check results.

4.1.6 Encryption Information

This is encrypted data and information that is associated with data for encryption purposes. Such information may include:

- encryption algorithm identifiers
- encryption keys.

4.1.7 Digital Signatures

Typically, a digital signature consists of encrypted integrity check information.

Where the encryption has been carried out using:

- a secret key that is known to a third party but not to the recipient

  or

- a private key in a public key encryption scheme, such that the information can be decrypted using the corresponding public key,

then a third party may subsequently be able to certify that the signature is genuine.

4.1.8 Certification Information

This is information supplied by a third party that certifies that another piece of information is genuine. For example, a certificate from a third party that verifies that a digital signature is genuine may be attached to the digital signature.

For certification information to be useful, the recipient must be able to verify that it was originated by the third party.

4.1.9 Sensitivity Labels

These carry an indication of the sensitivity of the information to which they are attached, as described in Section 2.2.7 on page 11.

4.2 Information Transmitted for the Network Infrastructure

4.2.1 Routing Information

This describes routes that must be taken, or must be avoided, for security routing control purposes. It is used by network nodes for routing purposes.

4.2.2 Dummy Traffic

Dummy traffic used for traffic padding is transmitted across networks but is not processed by communications partners.

4.3 Information that is not Transmitted

4.3.1 Encryption Information

Transmission of the following information is not essential to the operation of the security mechanisms. (For convenience, or other reasons, some of it may in some distributed systems be transmitted as *user data* by security applications.

Some encryption keys are not transmitted across networks but are distributed by other means (for example, by letter post in sealed envelopes).

Other encryption keys are transmitted between communications partners but, while in transit, they will normally be encrypted using secret keys or public keys. These secret keys, and the private keys corresponding to the public keys, are not transmitted.

4.3.2 Authorisation Information

This is information about the conditions under which an entity is authorised to access a resource. It may for example include lists of authorised users and groups of users, specifying the types of access that they are allowed. Such lists are called *access control lists* (abbreviated as *ACL*s). A more complete description of ACLs can be found in the referenced **X/Open Distributed Security Framework**.

4.3.3 Audit Trail Information

This is the information that is recorded for audit trail purposes.

4.3.4 Security Control Information

This includes:

- information that determines what events should be recorded for audit trail purposes
- information that determines what events should generate alarms
- information that determines what action should be taken when attempted security violations are detected.

Chapter 5

Security Services

The concept of a *security service* was introduced in Section 3.2 on page 16. The following set of security services are required to implement the security mechanisms listed in Section 2.2 on page 8.

- authentication
- authorisation
- integrity check
- encryption
- key distribution
- key certification
- alarm generation
- audit information recording.

X/Open does not currently define a security component whose purpose is to provide any of these services. However:

- X/Open does specify a programming interface, the GSS-API, that can be used to invoke some of them
- the X/Open DCE RPC and DCE Security components provide some of them, and could be used by other components
- the Directory Access component, in conjunction with an X.500 directory service, could provide two of them
- the Alarm Generation and Audit Information Recording services could be performed by Systems Management components.

5.1 The GSS-API

The Generic Security Service Application Programming Interface (GSS-API) is defined in the referenced **X/Open GSS-API** Preliminary Specification. This is based on the referenced **GSSAPI RFC**, which contains a programming-language-independent specification of the GSS-API, and the referenced **GSSAPI C Bindings RFC**, which contains a C language binding. A set of possible extensions to the GSS-API are discussed in the referenced **X/Open GSS-API Extensions** snapshot.

In addition to being specified in IETF RFCs and by X/Open, the GSS-API is planned to be supported by a future version of DCE and is being actively considered within ISO, POSIX and ECMA. It thus looks to have an important future as a security standard for open systems.

The GSS-API is intended to be used by software that implements communications protocols. In the context of the X/Open CAE, it would thus naturally be used by or within communications components. It could however also be used by applications in cases where the appropriate security is not provided by the communications components. This implies that the applications implement a security communications protocol.

The GSS-API provides an interface to a number of the services listed in Chapter 5 on page 25. These will be identified as the GSS-API and the proposed extensions are described.

5.1.1 GSS API Mechanisms

The means by which the services are provided, called *mechanisms* in the referenced **X/Open GSS-API** Preliminary Specification,[7] are not defined. The API can be used over a number of different mechanisms. For example, the GSS-API provides access to an encryption service. This service can be provided using secret key mechanisms or public key mechanisms. The choice of mechanism can be transparent to the software using the GSS-API.

5.1.2 Communications Protocols

Communication with communications partners is the responsibility of the programs that call the GSS-API, not of the programs that are invoked via the GSS-API.

The GSS-API can be used in conjunction with a number of different protocols. It assumes a message-oriented protocol rather than one that handles octet streams.[8] It can be used with connection-mode or connectionless message-oriented protocols. It might even (somewhat artificially) be used with a store-and-forward messaging protocol.

7. Elsewhere in this Technical Study, the term *mechanism* has the meaning given to it in the referenced **OSI Security Architecture** standard. The meaning here is somewhat different.

8. The integrity check and encryption facilities are applied to messages. They do not provide for the situation where the data is presented an octet at a time.

5.1.3 The Base GSS-API

The base GSS-API defined in the referenced **X/Open GSS-API** Preliminary Specification provides the following facilities.

Credential Management

Credentials are information structures that may contain identification, attribute and authentication information. The form of that information depends on the mechanisms that are used. The GSS-API enables a program to obtain a set of credentials and to dispose of them after use. The program can determine the name of the entity whose credentials they are, the lifetime for which they are valid, and the types of mechanism that are to be used in connection with them. With the base GSS-API, the rest of the information in the credentials is obtained from the environment, and the calling program has no control over it.

Context Management

A *security context* is an association between a pair of communications partners in which mutual authentication takes place, followed by communication of data that can be integrity-protected or encrypted. (Such an association is sometimes also called a *secure association*). The GSS-API enables a program to establish a security context with a communications partner and to terminate it after use.

Context establishment includes an authentication service. It uses credentials which may be obtained using the credential management facilities of the API. By default, they will be obtained by the service from the environment. The type of authentication employed depends on the mechanisms used to provide the service.

Context establishment can include access control if the mechanisms used support this. Access rights can be determined in the course of context establishment, and establishment of a context can be refused if the access rights are not sufficient. The GSS-API thus indirectly provides an interface to an authorisation service. Support for this means that the credentials must include all the identification and attribute information needed to determine access rights (this may, for example, include group identities as well as user identities), and it must also include the access type information.

Integrity Check and Encryption Facilities

Once a security context has been established, a program can use the GSS-API to create integrity check information for transmitted messages and to perform integrity checks on received messages. The checks can include checks against deletion, insertion or replay of messages, as well as checks against modification of messages. The API can also be used to encrypt transmitted data and to decrypt received data.

Supporting Facilities

The GSS-API provides a set of support facilities that:

- provide textual descriptions of status conditions
- enable callers to determine what types of mechanism are available
- manipulate internal representations of names
- release storage used for data structures allocated and returned by API functions.

Thus, the base GSS-API provides an interface to the following underlying services:

- authentication
- integrity check
- encryption.

It does not provide an interface to:

- key distribution
- key certification
- alarm generation
- audit information recording.

It also does not provide a direct interface to an authorisation service, but its context management functions indirectly provide authorisation for access control.

5.1.4 GSS-API Extensions

Provision of access control using the base GSS-API is not very satisfactory. The calling programs do not have sufficient ability to manipulate the information on which access control decisions are taken. This is especially a problem in the situation where a program communicates with a second program which in turn communicates with a third program that controls access to a resource, and the second program represents the first program in accessing the resource. In this case, the first program needs to delegate access rights to the second program. The base GSS-API does not provide for the construction of the appropriate credentials for the second program.

There are two main published proposals for extending the GSS-API to resolve this issue and to make other improvements. One of these is specific to the use of the GSS-API in connection with DCE. The other is generic. They enable programs that use the GSS-API to manipulate, within limits, information stored in credentials and other information associated with security contexts.

In the referenced **X/Open GSS-API Extensions** snapshot, these proposals are discussed and an interface definition is presented for a set of extensions to the GSS-API that follow the generic proposal.

5.2 DCE Security

The Distributed Computing Environment (DCE), defined by OSF and specified by X/Open as part of the X/Open Distributed Computing Services (XDCS) architecture, provides a coherent framework for distributed computing. Security was seen by the designers as an essential part of that framework, and security mechanisms form an integral part of DCE.

DCE incorporates a client-server paradigm in which client programs make remote procedure calls to server programs, as described in the referenced **DCE RPC** specification. These calls can be protected by the following mechanisms:

- authentication exchange
- access control
- data integrity
- encipherment
- digital signature.

As described in the referenced **DCE Security** specification, the following security services are provided by (unprotected) remote procedure call to enable the above mechanisms to be implemented:

- the Key Distribution Service (KDS), which provides certificated authentication and key distribution (it is also called the Authentication Service/Ticket Granting Service)
- the Privilege Service (PS), which provides certification of attribute information.

(Other security services - such as encryption - are also used, but they are provided locally within each client or server computer, rather than by a remote computer that acts as a security server.)

A *cell* is the basic unit of configuration and administration in the DCE architecture. Each cell contains a server that provides each of the remotely accessed security services. Protected remote procedure calls can be made within a single cell or between cells.

The DCE security services could be used to support communications paradigms other than remote procedure call. Such use is not currently covered by the DCE specifications. Other paradigms can only be supported provided that DCE RPC is also present, because of the role that DCE RPC plays in supporting the DCE security services. This applies not only within a single cell, but also between cells. If (for example) two cells were connected by a link over which process-to-process communication, but not remote procedure call, could be used, then DCE security could not be applied to protect communications across that link.

In addition, the DCE security services make essential use of the DCE time service, and could not be used in a system in which that service was not available.

The DCE security services can therefore provide comprehensive security in a set of networked systems that use DCE RPC exclusively, or that use it everywhere and use other communications services as well. On its own, it can not provide comprehensive security in all networked systems comprehended by the X/Open Distributed Computing Services architecture.

5.3 The X.500 Directory Service

The X.500 Directory Service is defined in the referenced **X.500** series CCITT recommendations. It is a distributed directory service, and it can hold information of various kinds. This can include encryption keys. These keys can be encrypted for secure distribution, and their origin can be certified, as described in CCITT Recommendation X.509. The X.500 Directory Service can thus provide a key distribution service and a key certification service.

The X/Open Directory Service API (XDS) provides an application interface to the X.500 Directory Service. The X/Open Directory Access component supports the XDS and uses the X.500 Directory Access Protocol (DAP) to communicate with an X.500 Directory System Agent (DSA). It thus provides part of the functionality of an X.500 Directory User Agent (DUA).

The DAP provides for authentication, integrity protection and encryption of the information that it carries. As currently specified, the XDS does not enable the program that invokes it to use the authentication, integrity protection and encryption services of the DAP. Therefore, if the key distribution and key certification services accessed via the XDS are to be secure, the information must be stored in the directory in authenticated and encrypted form. For information stored in accordance with X.509, this is the case.

The Directory Access component implementation must support the Strong Authentication Package (SAP) in order that the program invoking it can decrypt and verify the authenticity of information stored in the directory in accordance with X.509. The SAP is an optional feature of the XDS.

5.4 Management Services

The alarm generation service has to handle the raising of alarms and their subsequent cancellation once they have attracted attention. The audit information recording service has to handle:

- recording information about events at the API and the communications interface that are passed to it by the communications component
- enabling users (normally, these would be system managers or other users with special authorisation) to access that information and retrieve items from it
- preventing the recorded information from overflowing the bounds set by the storage medium.

These requirements are similar to those of certain systems management functions, such as performance management. It thus seems logical that audit information recording should be provided as a systems management function, as part of security management.

The provision of security alarm generation and audit information recording in this way requires further study.

Chapter 6

Security and Interworking APIs

This chapter discusses:

- the extent to which communications components for which there are X/Open interface specifications (see Section 1.2 on page 2) can support the security mechanism identified in Section 2.2 on page 8 against the threats discussed in Section 2.1 on page 6
- the extent to which those security mechanisms can be controlled via the APIs currently specified by X/Open.

As described in Section 3.2 on page 16, communications components can only implement security mechanisms to the extent that their communications interfaces can convey the security information that they require.

Each potential component can implement, to some extent, the security mechanisms that do not require security information to be conveyed by the communications protocol. These security mechanisms are:

- event detection
- audit trail
- security recovery.

Although these potential security mechanisms *can* be implemented, it should be noted that none of them are specified by, or even referred to in, X/Open interface specifications. None of them, therefore, can be controlled through X/Open APIs.

Ideally, the application of event detection and audit trail should be driven by the security-relevance of the events handled by a component. However, the extent to which they can be implemented by communications components partly depends on the extent to which other security mechanisms can be implemented, since it is the other security mechanisms that enable attempted security violations to be detected and recorded for audit.

The situation for the security mechanisms other than event detection, audit trail and security recovery is presented in the following tables. They are of two kinds:

- the security information tables have three columns, giving:
  - — the X/Open interworking specifications
  - — the protocols that they reference
  - — the security information that those protocols are capable of conveying.
- the security mechanisms tables have four columns, giving:
  - — the X/Open interworking specifications
  - — the security mechanisms that could be applied based on the security information conveyed by the protocols that they reference
  - — an indication of whether those security mechanisms are described in the specification and, if they are,
  - — an indication of whether they can be controlled via the specified API.

The X/Open interworking specifications that are considered include all of the specifications that are identified in the referenced **XDCS** Framework specification. The communications interfaces that are considered are all those that are referenced in the specifications that are considered. They include all those that are identified in the referenced **XDCS** Framework specification.

X/Open Specification	**Protocol**	**Security Information**
XTI	ISO COTP	None
XTI	ISO CLTP	None
XTI	mOSI	None
XTI	TCP	None
XTI	UDP	None
XTI	RFC 1006	None
XTI	IP	None
XTI	NETBIOS	None[1]
XSockets	TCP	None
XSockets	UDP	None
XAP	ISO ACSE (+ Presentation and Session)	All[2]
IPC for SMB	SMB	Identification Authentication

Notes:

1 The NETBIOS naming mechanism is here considered to be a protocol addressing mechanism rather than a logical identification mechanism.

2 The OSI ACSE provides for authentication information to be exchanged during association establishment (see the referenced **ACSE Authentication** amendment to the ACSE standard, but note that a defect to the definition of Authentication-value in the associated protocol specification has been reported, and a change to this definition is anticipated).

The OSI presentation service provides for the conveyance of information of all kinds, including security information. This information is provided and interpreted by the users of the XAP API. It is transparent to implementations of the XAP API.

Table 6-1 Process-to-Process Communication - Information

X/Open Specification	Security Mechanism	Specification	Control
XTI	None		
XSockets	None		
XAP	None[1]		
IPC for SMB	Authentication	YES	NO
	Access	YES	NO

Notes:

1 The OSI ACSE provides for authentication information to be exchanged during association establishment (see the referenced **ACSE Authentication** amendment to the ACSE standard, but note that a defect to the definition of Authentication-value in the associated protocol specification has been reported, and a change to this definition is anticipated). The XAP API does not give the application access to this information.

The OSI presentation service provides for the conveyance of information of all kinds, including security information. This information is provided and interpreted by the users of the XAP API. It is transparent to implementations of the XAP API.

Table 6-2 Process-to-Process Communication - Security Mechanisms

X/Open Specification	**Protocol**	**Security Information**
XNFS	ONC RPC	Identification Attribute Authentication[1]
(PC) NFS	ONC RPC	Identification Attribute Authentication[1]
DCE RPC	DCE RPC	Identification Attribute Authentication Access Type Integrity Encryption Digital Signature Certification

Notes:

1 In the referenced **XNFS** CAE Specification and the referenced **(PC) NFS** Preliminary Specification, a general mechanism is defined for passing authentication, but the specific authentication schemes that are defined do not use it; one provides null authentication, the other authenticates simply on a match of the identification information.

Table 6-3 Remote Procedure Call - Information

X/Open Specification	Security Mechanism	Specification	Control
XNFS	Authentication	N/A[1]	
(PC) NFS	Authentication	N/A[1]	
DCE RPC	Authentication	YES	YES
	Access	YES	YES
	Integrity	YES	YES
	Encypher	YES	YES
	Dig Sign	YES[2]	YES
	Notarisation	YES[3]	YES

Notes:

1 No API is defined in the referenced **XNFS** CAE Specification or the referenced **(PC) NFS** Preliminary Specification.

2 User data is not digitally signed in a way that allows notarisation.

3 Only of certain security information, not of user data.

Table 6-4 Remote Procedure Call - Security Mechanisms

X/Open Specification	Protocol	Security Information
XFTAM	FTAM	Identification
		Authentication
		Access Type
BSFT	FTAM	Identification
		Authentication
		Access Type
XDCS	FTP	Identification
		Authentication
		Access Type
XNFS	NFS	Identification[1]
(PC) NFS	NFS	Identification[1]

Notes:

1 NFS also uses the identification and authentication information conveyed by ONC RPC.

Table 6-5 File Management - Information

X/Open Specification	Security Mechanism	Specification	Control
XFTAM	Authentication	YES	YES
	Access	YES	YES
BSFT	Authentication	YES	N/A[1]
	Access	YES	N/A[1]
XDCS(FTP)	Authentication	YES	N/A[1]
	Access	YES	N/A[1]
XNFS	Access[2]	YES	N/A[3]
(PC) NFS	Access[2]	YES	N/A[3]

Notes:

1 The interfaces defined in the referenced **FTP** RFC and the referenced **BSFT** CAE Specification are user interfaces, not APIs.

2 Authentication is provided by the underlying ONC RPC protocol.

3 No API is defined in the referenced **XNFS** CAE Specification or the referenced **(PC) NFS** Preliminary Specification.

Table 6-6 File Management - Security Mechanisms

X/Open Specification	Protocol	Security Information
XMHS	X.400 (1988) P1	Identification Authentication Integrity Encryption Digital Signature Certification Label
XMHS	X.400 (1988) P2	Identification Encryption[1] Label
XMHS	X.400 P3	Identification Authentication Integrity Encryption Digital Signature Certification Label
XMHS	X.400 P7	Identification Authentication Integrity Encryption Digital Signature Certification Label
XMHS	SMTP[2]	Identification

Notes:

1 CCITT Recommendation X.420 (1988) defines an *Encrypted Body Part*, but states that its contents are for further study.

2 No mapping to SMTP is defined in the referenced **XMHS** CAE specification, but a mapping is possible.

Table 6-7 Message Handling - Information

X/Open Specification	Security Mechanism	Specification	Control
XMHS	Authentication	YES	YES
	Integrity	YES	YES
	Encypher	YES	YES
	Dig Sign	YES	YES
	Notarisation	YES	YES
	Label	YES	YES

Table 6-8 Message Handling - Security Mechanisms

X/Open Specification	Protocol	Security Information
XDS	X.500 DAP	Identification
		Authentication
		Integrity
		Digital Signature
		Certification
XDS	DNS[1]	None
DCE Directory	DCE CDS	None[2]

Notes:

1 No specific mapping of DNS protocols to the XDS API is defined in the referenced **XDS** CAE Specification, but such a mapping is possible.

2 The DCE CDS protocols use authenticated RPC as an underlying protocol.

Table 6-9 Directory - Information

X/Open Specification	Security Mechanism	Specification	Control
XDS	Authentication	YES	NO[1]
	Access	YES	YES
	Integrity	YES	NO[2]
	Dig Sign	YES	NO[1,2]
	Notarisation	YES	NO[1]
DCE CDS	None[3]		[4]

Notes:

1 The directory can be used to store authentication information, and the XDS can be used to access the information that is stored in the directory. Integrity protection and digital signatures of authentication information stored in the directory are supported. This table entry, however, is concerned with control of the security mechanism when the user accesses *any* information stored in the directory. The DAP provides for notarised mutual authentication information to be exchanged. The XDS as defined in the referenced **XDS** CAE Specification does not enable the application to provide or access this information. Note, however, that DCE uses XDS as the API to its global directory service. For this purpose, an extension of XDS is defined in the referenced **DCE Directory** Preliminary Specification. This extension allows the application to specify authentication information, in the form of a password.

2 The DAP provides for directory access data units to be integrity protected and signed. The XDS does not enable the application to use these features of the DAP.

3 The DCE CDS protocols use authenticated RPC as an underlying protocol.

4 The XDS is the only API specified in the referenced **DCE Directory** Preliminary Specification. The security information available through the underlying RPC API is not available through the XDS API.

Table 6-10 Directory - Security Mechanisms

X/Open Specification	Protocol	Security Information
XMP	CMIP	Unspecified[1]
XMP	SNMP	Identification Unspecified[2]

Notes:

1 The protocol data units defined in the referenced **CMIP** standard include OPTIONAL EXTERNAL access control fields.

2 SNMP does not specify formats for data unit containing security information, but it does allow implementations to define their own formats for data units, and explains that authentication can be provided in this way. Implementations that do this are unlikely to interwork with other implementations.

Table 6-11 Systems Management - Information

X/Open Specification	Security Mechanism	Specification	Control
XMP	Unspecified	YES[1]	YES[1]

Notes:

1 Provision is made in the referenced **XTI** CAE Specification for the access control fields referred to in the referenced **CMIP** standard and for implementation-defined security information in SNMP.

Table 6-12 Systems Management - Security Mechanisms

X/Open Specification	Protocol	Security Information
CPI_C	OSI-TP	Identification[1]
XATMI	OSI-TP	Identification[1]
TxRPC	TxRPC[2]	Identification Authentication
CPI-C	LU 6.2	Identification Authentication

Notes:

1 In the referenced **OSI TP** standard it states that the provision of security is a candidate for further standardisation.

2 The TxRPC specification defines a protocol stack that includes an OSI stack (up to Presentation layer), the DCE RPC protocol and the OSI TP protocol. This stack does not make use of the ability of the DCE RPC protocol to convey security information, but it does include a separate ability (defined in the TxRPC specification) to carry identification and authentication information. This information - **Client-Authenticator** - does not support mutual authentication.

Table 6-13 Transaction Processing - Information

X/Open Specification	Security Mechanism	Specification	Control
XATMI	None		
TxRPC	Authentication Xchg	YES	YES
CPI-C	Authentication Xchg	YES	YES

Table 6-14 Transaction Processing - Security Mechanisms

X/Open Specification	Protocol	Security Information
SQL	RDA	Identification Authentication Access Type
CLI	RDA	Identification Authentication Access Type

Table 6-15 Data Management - Information

X/Open Specification	Security Mechanism	Specification	Control
SQL	Authentication	YES	YES
	Access	YES	NO[1]
CLI	Authentication	YES	YES
	Access	YES	NO[1]

Notes:

1 Particular SQL statements or API function calls imply particular access types, but the *specificUsageMode* parameter of the *R-Open* service is not visible at the SQL or CLI interface.

Table 6-16 Data Management - Security Mechanisms

X/Open Specification	Protocol	Security Information
XDCS	telnet	Identification Authentication
Xlib	X/Windows	Identification[1] Authentication[1]

Notes:

1 In the referenced **X Protocol** CAE Specification, a mechanism is defined for the client to send the server the name of an authorisation protocol and authorisation information, but the protocols that can be used and the information that they require are not defined.

Table 6-17 Terminal Access - Information

X/Open Specification	Security Mechanism	Specification	Control
XDCS(telnet)	Authentication Access	YES YES	1 1
Xlib	Authentication Access	NO NO[2]	

Notes:

1 No API is defined in the referenced **Telnet** RFC.

2 The authorisation information referred to in the referenced **X Protocol** CAE Specification is not exposed in the interface defined in the referenced **Xlib** CAE Specification, but that interface provides an alternative means of access control based on host identity.

Table 6-18 Terminal Access - Security Mechanisms

Chapter 7

New Protocols

This section briefly discussed some protocols that support interworking security but that are not currently referenced by X/Open specifications.

7.1 OSI Secure Network and Transport Protocols

The OSI Network Layer Security Protocol (NLSP) is defined in the referenced **NLSP** standard. It provides both a connection-mode service and a connectionless-mode service. For either type of service, it can convey the following information:

- identification information
- authentication information
- attribute information (in the *authentication information* fields of the protocol data units)
- access type information (in the *authentication information* and *security label* fields of the protocol data units)
- integrity information
- encryption information
- digital signatures (as integrity information or encryption information)
- certification information (in the *authentication information* fields of the protocol data units)
- sensitivity labels (in the *security label* fields of the protocol data units)
- dummy traffic.

This information can support the following mechanisms:

- authentication exchange
- access control
- data integrity
- encipherment (of user data and of protocol information)
- digital signature (as a by-product of data integrity or encipherment)
- security labelling
- routing control (by sensitivity labels)
- traffic padding.

The form of the information is not specified, and the algorithms to be used to implement the mechanisms are not defined. They can be determined by bilateral agreement or implied by the *SA-P Type* field of the protocol's Security Association data unit. This field contains an ASN.1 object identifier that indicates the security association protocol to be used. An annex to the referenced **NLSP** standard defines one security association protocol that can be used as part of the NLSP.

The OSI Transport Layer Security Protocol (TLSP) is defined in the referenced **TLSP** standard and the referenced **OSI Transport Security Association** draft ammendment. It is in many ways similar to the NLSP. The differences are that the TLSP does not support traffic padding or encryption of network addressing information, and that the protocols operate at different layers of the OSI reference model.

The NLSP operates at the network layer, and the TLSP operates at the transport layer. Each has particular advantages and disadvantages.

Operation at the network layer means that traffic padding and encryption of network address information can be provided, giving protection against traffic analysis. However, several transport connections can be carried over a single network connection; this may lead to sensitive data on one transport connection being encrypted together with other data on other transport connections, and this may present opportunities for cryptographic attack if the attacker can gain access to the other data. This danger does not arise with operation at the transport layer before the procedures of multiplexing and assignment to network connection are carried out, as is the case with TLSP.

It should be noted that encryption of network addresses with NLSP means that NLSP must be implemented by the intermediate systems through which the traffic passes, as well as in the end systems containing the sending and receiving applications programs.

Because they are so similar in function, it is unlikely that the NLSP and the TLSP would be used together.

7.2 OSI Generic Upper Layer Security

While the OSI ACSE and Presentation protocols are referenced in XAP, this does not cover their use to convey security information, which is still being standardised.

The conveyance of security information by OSI upper layer (presentation and application layer) protocols is defined in the referenced **OSI Upper Layer Security Model** draft standard and the referenced **GULS** draft standards. They provide for:

- exchanges of security information by applications (this can include all of the types of information listed in Section 4.1 on page 22)
- the conveyance by the presentation layer (and the definition of presentation contexts for) data that has been subjected to *security transformations* such as encryption, addition of integrity protection information and addition of digital signatures.

7.3 Internet Protocols

Work towards defining new Internet protocols with security features includes the following:

- **RFCs 1421, 1422, 1423 and 1424** specify secure electronic mail facilities that are compatible with a number of mail protocols, including SMTP
- **RFC 1281** gives guidelines for secure operation of the Internet
- **RFC 1321** specifies the MD5 Message-Digest Algorithm proposed as the preferred Internet algorithm by IPng (Internet Protocol next generation)
- **RFC 1352** specifies security mechanisms for use with SNMP
- **RFCs 1441, 1446 and 1448** describe the SNMP version 2 Framework and protocols, which include provision for authentication, access control, data integrity and encryption
- **RFCs 1411, 1412 and 1416** describe authentication schemes for Telnet

- **RFC 1413** defines an identification protocol for TCP
- **RFC 1457** describes a security labelling framework
- **RFC 1507** describes a distributed authentication service based on public key encryption
- **RFC 1510** describes the Kerberos authentication service based on secret key encryption.
- **IPng (or IPv6)** as currently specified, this provides optional authentication and integrity to users by way of an optional authentication header.

Chapter 8

Conclusions

There are a number of threats that arise due to the presence of communications components in a system. There is a set of security mechanisms, that can be implemented in communications components, and that provide some protection against these threats. For more complete protection, they must be supplemented by countermeasures provided by other means such as network design and systems management. They are described in Chapter 2.

The sets of threats and security mechanisms may not cover all the possibilities, but they reflect the best available technical opinion on what is needed. That opinion has now been stable for some time.

For the security mechanisms to be implemented, the communications protocols must be capable of conveying certain security information. The information required for the security mechanisms is identified in Chapter 4. The extent to which the communications protocols referred to in X/Open specifications can convey this information is summarised in the tables in Chapter 6. Except for X.400, X.500 and DCE RPC, the protocols are not able to convey as much security information as is desirable.

The inability to convey security information is of particular concern in the case of protocols used for systems management, since systems can be very vulnerable to attacks that make improper use of management facilities.

There are some new protocols being defined that can convey security information. These are discussed in Chapter 7.

The application of security mechanisms by communications components can be controlled in several ways. The most important are: control through an API by security-aware applications, and control through systems management facilities. Control through systems management facilities is expected to be the primary requirement.

The extent to which the X/Open communications APIs support control by applications is summarised in the tables in Chapter 6. With a few exceptions (notably, that of XDS) they enable applications to control the security mechanisms that are supported by the information conveyed by the communications protocols. They do not enable the applications to control other security mechanisms such as Audit Trail.

The development of a common API that would provide for control of security mechanisms in all communications components should be considered. This would provide a means for systems management facilities or applications to control security mechanisms that can be provided by communications components but that can not be controlled through the existing interworking APIs.

Implementation of the security mechanisms by communications components requires those components to use a number of underlying security services. They are discussed in Chapter 5 on page 25. X/Open does not specify whether these underlying security services are to be incorporated within the communications components or to be provided by separate security components.

X/Open does however specify a programming interface to generic security services. This is the GSS-API. It covers all of the required services except key distribution, key certification, alarm generation and audit information recording. At present, it does not handle authorisation very well, but it is being enhanced to correct this.

Key distribution and key certification services can be provided by the X/Open Directory Access component in conjunction with an X.500 directory service.

Audit information recording and alarm generation can be provided by systems management services. The appropriate services are currently not specified by X/Open.

All of the underlying security services except audit information recording and alarm generation are provided by X/Open DCE RPC and the X/Open DCE Security Service. The DCE security architecture is comprehensive within the context of DCE, but it can not meet all the needs for underlying security services to support interworking within the X/Open CAE, because it can not be used by systems that do not support DCE RPC.

The elements required to provide interworking security within the X/Open CAE are:

- communications protocols that convey security information
- API facilities that provide control of security mechanisms (possibly including a common security mechanisms control API)
- security-related feature options, configuration parameters and environment variables
- systems management functions that control security mechanisms
- underlying services, provided by a combination of:
  - — implementations of the GSS-API
  - — X/Open DCE RPC and the X/Open DCE Security Service
  - — the X/Open Directory Access component and an X.500 directory service
  - — systems management functions.

This Technical Study describes the extent to which these elements are currently realised. Work on them continues.

Appendix A

XTI Security Considerations

This appendix discusses the security aspects of the XTI API. Its purposes are to identify the security requirements associated with the API, to describe the existing security mechanisms that the API provides, and to identify actions that may enable security of systems that include the XTI API to be improved. It is intended that this material will be included, with some changes (see below), in the the referenced **XTI** CAE Specification.

In this version, the final section of this appendix (see Section A.3 on page 57) contains a number of specific proposals to be decided by X/Open. This material will be revised to reflect these decisions when it is included in the the referenced **XTI** CAE Specification.

A.1 Security Issues

A.1.1 Threats

Interworking Threats

All of the threats described in the body of this technical study apply to systems that include components that support the referenced **XTI** CAE Specification. These threats are:

- **unauthorised modification** resulting from:
  - masquerade of a communications partner
  - tampering with communications media
  - security breach in a communications partner
- **unauthorised disclosure** resulting from:
  - masquerade of a communications partner
  - monitoring of or tampering with communications media
  - traffic analysis
  - security breach in a communications partner
- **unauthorised use of resources** resulting from:
  - masquerade of a communications partner
  - an unauthorised user presenting himself as though he were authorised
- **denial of service** resulting from:
  - damage to communications networks
  - congestion in communications networks
- **repudiation** resulting from:
  - the sender of information denying that it was sent in the form that it was received
  - the receiver of information denying that it was received in the form that it was sent.

API Threats

In addition, the following threats that are associated with APIs in general apply to the XTI API:

- **unauthorised modification** resulting from:
  - masquerade of a user of the API
  - access by another process to memory used by the API implementation (this is a particular issue since the XTI provides the application with access to memory via *t_alloc*( ))
- **unauthorised disclosure** resulting from:
  - masquerade of a user of the API
  - access by another process to memory used by the API implementation (this is a particular issue since the XTI provides the application with access to memory via *t_alloc*( ))

- **unauthorised use of resources** resulting from:
  — masquerade of a user of the API
  — an unauthorised user of the API presenting himself as though he were authorised
- **denial of service** resulting from:
  — unavailability of processing resources (for example, memory).

Other Threats

The XTI is often implemented over other libraries which may have their own APIs. In such a case, there is a possibility of one of the API threats listed above being realised through the API to the underlying library. Unless the API to the underlying library is secure enough to prevent these threats being realised, an application could circumvent security mechanisms implemented for the XTI by using the API to the underlying library.

Since no API to an underlying library is specified or assumed in the referenced **XTI** CAE Specification, the discussion of specific countermeasures to security breaches in an API to an underlying library is outside the scope of this Technical Study. However, implementors and purchasers of components supporting the XTI should be aware of the possibilities.

Also, in some implementations, the XTI is partly implemented in user space. This means that some of the memory used internally by the implementation may be accessible by the application. There are a number of threats associated with this possibility. In particular, they include unauthorised disclosure and modification. Again, since the question of whether the XTI is partly implemented in user space is not covered in the referenced **XTI** CAE Specification, the discussion of specific countermeasures to these threats is outside the scope of this Technical Study, but implementors and purchasers of components supporting the XTI should be aware of them.

A.1.2 Policy for Security Implementation

The recommendations in this appendix are based on the following policy:

- there are certain threats that are not appropriate to be countered by implementations of the XTI. These are:
  — threats arising due to security breaches in communications partners (these should be addressed by ensuring that communications partners are sufficiently trustworthy)
  — threats arising due to damage to or congestion in networks (these should be addressed by network design and management practices)
  — threats of repudiation (these should be addressed by those applications of the XTI that are subject to them)
  — threats arising due to masquerade of a user of the XTI API or to a masquerade to a user of an implementation of the XTI (these should be addressed by other parts of the environment at logon time or when a process attempts to change its *uid*)
- ideally, components supporting the referenced **XTI** CAE Specification should include countermeasures to the other threats
- it should be possible for systems management programs and security-aware applications to control these security mechanisms
- the most appropriate means of control would be a common security mechanisms control API

- for most applications, threats of denial of service are less serious than other threats
- most current implementations of XTI do not support security mechanisms. Any enhancements to the XTI should be made optional, so that existing implementations will continue to conform
- the protocols referenced in the referenced **XTI** CAE Specification do not convey the security information necessary to support security mechanisms. No events that would justify raising alarms are detected. The only protection XTI can provide in practice, when used with these protocols, is audit trail. Support for the OSI secure network and transport protocols should be added, so that the XTI can provide a secure means of process-to-process communication for those applications that require it.

A.1.3 Impact on Other Specifications

If control of security mechanisms through systems management facilities is to be provided, there may be an impact on systems management specifications.

A.2 Overview of Security Facilities Provided

A.2.1 Security Goals

Implementations of the XTI should optionally provide applications with a process-to-process communications service that is secure to the extent that this is permitted by the communications protocol that is used.

It should be possible to use the XTI in conjunction with protocols that allow it to provide a secure service.

A.2.2 Security Framework

See the referenced **X/Open Distributed Security Framework** and the body of this technical study.

A.2.3 Security Functionality and Services

All the X/Open security classes defined in the referenced **Secure Systems Procurement** X/Open Guide are potentially applicable to an implementation of the XTI and the applications that use it. As the XTI API supports the implementation of distributed applications and systems, the services that are grouped under the X-DIST security class are likely to be of particular relevance to XTI applications.

A.2.4 Security Information

Use of the XTI API in conjunction with the following protocols is defined in the referenced **XTI** CAE Specification:

- the ISO connection-mode transport protocol defined in the referenced **OSI COTS** standard
- the ISO connectionless-mode transport protocol defined in the referenced **OSI CLTS** standard
- the TCP protocol defined in the referenced **TCP** RFC
- the UDP protocol defined in the referenced **UDP** RFC
- the IP protocol defined in the referenced **IP** RFC
- the NETBIOS protocol defined in the referenced **NETBIOS** specification.

In addition, there are agreed (but not yet published) new appendices for the following protocols:

- the COTS protocol over TCP as defined in the referenced **COTS-over-TCP** RFC
- the minimal 7-layer OSI stack defined in the referenced **Minimum OSI** profile
- the X.25 protocol defined in the referenced **X.25** CCITT recommendation.

None of these protocols make any explicit provision for conveying security information.

A.2.5 Standards

The security services and mechanisms for open systems interworking are defined in the referenced **OSI Security Architecture** standard.

The OSI secure transport protocol is defined in the referenced **TLSP** standard.

The OSI secure network protocol is defined in the referenced **NLSP** standard.

A.2.6 Emerging Standards

An addition to the OSI secure transport protocol is defined in the referenced **OSI Transport Security Association** draft ammendment.

A.3 Security Specification

The following are specific conclusions in the light of the policies stated in Section A.1.2 on page 53.

1. Support for the audit trail security mechanism could be specified as an optional feature of the XTI. However, it would be preferable not to do this. Rather, X/Open should make it possible for component definitions to require audit trail to be provided in conjunction with XTI.

   The interworking and systems management experts within the X/Open organisation should liaise to discuss a security API that would provide for control of the audit trail security mechanism and that would be applicable to components that support the XTI.

   The events recorded should be normal protocol events (connections, disconnections etc.) and API calls.

2. Support for the OSI Network Layer Security Protocol (NLSP) and Transport Layer Security Protocol (TLSP) could be described in appendices to the XTI specification.

   The NLSP would normally be implemented in a protocol stack that used the normal OSI Transport Protocol (connection-mode or connectionless-mode as appropriate) at the transport layer. Support for NLSP could be added to XTI by defining a new set of options to be used with the OSI Transport protocol; these options would select the NLSP and provide control over its use by allowing the application to read and set security information.

   Use of the TLSP with XTI could be similar to use of the normal OSI transport protocol, but again a new set of options would be required to enable applications to control its operation.

   These issues should be addressed by X/Open at the appropriate time with regard to the maturity of the OSI NLSP and TLSP standards.

3. There are security threats associated with *t_alloc*( ). To counter these threats, an implementation should ensure that:

   - one user can not access memory that has been allocated to another user
   - the contents of memory that has been allocated and returned to store will be erased before the memory is re-used
   - there is a limit on the proportion of available memory that can be allocated to a single user.

   In any implementation, *t_alloc*( ) is likely to behave in these respects in the same way as *malloc*( ) (although this can not be guaranteed).

   A prospective purchaser of components supporting the XTI should be aware of the security threats associated with *t_alloc*( ), and should ensure that the implementation purchased provides the appropriate degree of security.

4. The XTI as currently specified provides for return of the TACCES error in case of access control violations, but it does not define the precise circumstances in which this error will be generated. Further work is needed to determine the circumstances in which access to a resource should be denied to a user of the XTI.

   The XNET group of X/Open should liaise with the Security Working Group of X/Open to determine the best way of defining appropriate access control mechanisms (such as the access control lists - ACLs - described by the POSIX P1003.1e security working group) and should ensure that they are described, either in the XTI specification or in another specification that the XTI specification can reference.

5. *Secure XTI* feature options could be defined for components that include the XTI. An implementation claiming support for these options could be required to:

   - support the audit trail security mechanism as in 1
   - support the OSI NLSP and TLSP as in 2
   - support access control as in 4
   - support memory separation for *t_alloc*( )
   - support allocation restrictions for *t_alloc*( ).

The relevant interworking and security areas of expertise within the X/Open organisation have to decide:

- the options that would be appropriate

  and

- the mechanisms that should be required for each option.

Appendix B

DCE RPC Security Considerations

This appendix discusses the security aspects of DCE RPC. Its purposes are to identify the security requirements associated with DCE RPC, to describe the existing security mechanisms that DCE RPC provides, and to identify actions that may enable security of systems that include DCE RPC to be improved. It is intended that this material will be included, with some changes (see below), in the referenced **DCE RPC** specification.

In this version, the final section of this appendix (see Section B.3 on page 63) contains a number of specific proposals to be decided by X/Open. This material will be revised to reflect these decisions when it is included in the referenced **DCE RPC** specification.

B.1 Security Issues

B.1.1 Threats

Interworking Threats

All of the threats described in the body of this technical study apply to systems that include components that support the referenced **DCE RPC** specification. These threats are:

- **unauthorised modification** resulting from:
  - — masquerade of a communications partner
  - — tampering with communications media
  - — security breach in a communications partner
- **unauthorised disclosure** resulting from:
  - — masquerade of a communications partner
  - — monitoring of or tampering with communications media
  - — traffic analysis
  - — security breach in a communications partner
- **unauthorised use of resources** resulting from:
  - — masquerade of a communications partner
  - — an unauthorised user presenting himself as though he were authorised
- **denial of service** resulting from:
  - — damage to communications networks
  - — congestion in communications networks
- **repudiation** resulting from:
  - — the sender of information denying that it was sent in the form that it was received
  - — the receiver of information denying that it was received in the form that it was sent.

API Threats

In addition, the following threats that are associated with APIs in general apply to the DCE RPC API:

- **unauthorised modification** resulting from:
  - — masquerade of a user of the API
  - — access by another process to memory allocated by the API implementation (for DCE RPC, memory is allocated by stub memory management to store arguments of some remote procedures)
- **unauthorised disclosure** resulting from:
  - — masquerade of a user of the API
  - — access by another process to memory allocated by the API implementation

- **unauthorised use of resources** resulting from:
  - — masquerade of a user of the API
  - — an unauthorised user of the API presenting himself as though he were authorised
- **denial of service** resulting from:
  - — unavailability of processing resources (for example, memory).

Other Threats

No other threats associated with the DCE RPC have been identified.

B.1.2 Policy for Security Implementation

The recommendations in this appendix are based on the following policy.

- There are certain threats that are not appropriate to be countered by implementations of the DCE RPC. These are:
  - — threats arising due to security breaches in communications partners (these should be addressed by ensuring that communications partners are sufficiently trustworthy)
  - — threats arising due to damage to or congestion in networks (these should be addressed by network design and management practices)
  - — threats of disclosure through traffic analysis (these will not apply to most users of the DCE RPC - those users to which they do apply must address them by application-specific means, or by ensuring that the protocol stack underlying the RPC protocol provides protection, for example by including the OSI NLSP)
  - — threats of repudiation (these should be addressed by those applications of the DCE RPC API that are subject to them).
  - — threats arising due to masquerade of a user of the DCE RPC API or to a masquerade to a user of an implementation of the DCE RPC API (these should be addressed by other parts of the environment at logon time or when a process attempts to change its *uid*).
- For most applications, threats of denial of service are less serious than other threats.
- The DCE RPC specification currently defines a wide range of security mechanisms. They are provided by existing implementations, and are adequate for most purposes. Any enhancements should be made optional, so that existing implementations will continue to conform.
- The most appropriate means of control of any additional security mechanisms would be a common security security mechanisms control API.

B.1.3 Impact on Other Specifications

If control of security mechanisms through systems management facilities is to be provided, there may be an impact on systems management specifications.

B.2 Overview of Security Facilities Provided

B.2.1 Security Goals

Implementations of the DCE RPC should provide applications with a secure remote procedure call service.

B.2.2 Security Framework

See the referenced **X/Open Distributed Security Framework**, the referenced **DCE Security** specification and the body of this technical study.

B.2.3 Security Functionality and Services

All the X/Open security classes defined in the referenced **Secure Systems Procurement** X/Open Guide are potentially applicable to an implementation of the DCE RPC and the applications that use it. As the DCE RPC API supports the implementation of distributed applications and systems, the services that are grouped under the X-DIST security class are likely to be of particular relevance to DCE RPC applications.

B.2.4 Security Information

DCE RPC uses RPC protocols defined in the referenced **DCE RPC** specification. These protocols make explicit provision for conveying the following types of security information:

- identification
- authentication
- attribute
- access Type
- integrity
- encryption
- digital Signatures
- certification.

B.2.5 Standards

None applicable.

B.2.6 Emerging Standards

None applicable.

B.3 Security Specification

At present, the referenced **DCE RPC** specification provides support for the following security mechanisms:

- authentication exchange
- access control
- data integrity
- encipherment
- digital signature
- notarisation of some security information, but not in general of user data.

These security mechanisms provide specific protection against all of the interworking threats for which security mechanisms are required by the security implementation policy stated in Section B.1.2 on page 61. They do not, however, include general interworking security mechanisms or security mechanisms against API threats.

The following are specific conclusions in the light of the policies stated in Section B.1.2:

1. Support for the audit trail security mechanism and for alarm generation following event detection could be specified in the DCE RPC as optional features. However, it would be preferable not to do this. Rather, X/Open should make it possible for component definitions to require audit trail and alarm generation to be provided in conjunction with DCE RPC.

   The events recorded for audit should be normal security-related events (requests for tickets, etc.) plus attempted security violations such as authentication failures.

   Attempted security violations could result in generation of alarms.

   The interworking and systems management experts within the X/Open organisation should liaise to discuss a security API that would provide for control of the audit trail and alarm generation security mechanisms and that would be applicable to components that support DCE RPC.

2. There are security threats associated with stub memory management. To counter these threats, an implementation should ensure that:

   - other processes are prevented from accessing the memory that is allocated to a process by stub memory management
   - the contents of memory that has been allocated and returned to store are erased before the memory is re-used
   - there is a limit on the proportion of available memory that can be allocated to a single user.

   A prospective purchaser of components supporting DCE RPC should be aware of the security threats associated with stub memory management and should ensure that the implementation purchased provides the appropriate degree of security.

The relevant interworking and security areas of expertise within the X/Open organisation have to decide:

- the options that would be appropriate

  and

- the mechanisms that should be required for each option.

Appendix C

XDS Security Considerations

This appendix discusses the security aspects of the XDS API. Its purposes are to identify the security requirements associated with the API, to describe the existing security mechanisms that the API provides, and to identify actions that may enable security of systems that include the XDS API to be improved. It is intended that this material will be included, with some changes (see below), in the the referenced **XDS** CAE Specification.

In this version, the final section of this appendix (see Section C.3 on page 71), contains a number of specific proposals to be decided by X/Open. This material will be revised to reflect these decisions when it is included in the the referenced **XDS** CAE Specification.

C.1 Security Issues

C.1.1 Threats

It should be noted that the directory service can be used to provide access to security information, including certificates of authentication and encryption keys. A security breach due to the realisation of threats listed in this section could lead to security breaches in other areas. For example, a successful masquerade attack via the XDS could enable the attacker to obtain credentials that would allow him to masquerade as a privileged user of applications provided by systems on his network, and to access or modify the information maintained by those applications. The possibilities for theft of money (with financial applications), prejudice to national interests (with government applications) etc. are obvious.

Interworking Threats

All of the threats described in the body of this technical study apply to systems that include components that support the referenced **XDS** CAE Specification. These threats are:

- **unauthorised modification** resulting from:
  — masquerade of a communications partner
  — tampering with communications media
  — security breach in a communications partner
- **unauthorised disclosure** resulting from:
  — masquerade of a communications partner
  — monitoring of or tampering with communications media
  — traffic analysis
  — security breach in a communications partner
- **unauthorised use of resources** resulting from:
  — masquerade of a communications partner
  — an unauthorised user presenting himself as though he were authorised
- **denial of service** resulting from:
  — damage to communications networks
  — congestion in communications networks
- **repudiation** resulting from:
  — the sender of information denying that it was sent in the form that it was received
  — the receiver of information denying that it was received in the form that it was sent.

API Threats

In addition, the following threats that are associated with APIs in general apply to the XDS API:

- **unauthorised modification** resulting from:
  - — masquerade of a user of the API
  - — access by another process to memory used by the API implementation to store private objects and service-generated public objects
- **unauthorised disclosure** resulting from:
  - — masquerade of a user of the API
  - — access by another process to memory used by the API implementation to store private objects and service-generated public objects
- **unauthorised use of resources** resulting from:
  - — masquerade of a user of the API
  - — an unauthorised user of the API presenting himself as though he were authorised
- **denial of service** resulting from:
  - — unavailability of processing resources (for example, memory).

Other Threats

No other threats associated with the XDS have been identified.

C.1.2 Policy for Security Implementation

The recommendations in this appendix are based on the following policy.

- There are certain threats that are not appropriate to be countered by implementations of the XDS. These are:
  - — threats arising due to security breaches in communications partners (the communications partner of an XDS application is the directory service - the security policy of the directory service user must insist that this is sufficiently trustworthy)
  - — threats arising due to damage to or congestion in networks (these should be addressed by network design and management practices)
  - — threats of disclosure through traffic analysis (these will not apply to most users of the XDS - those users to which they do apply must address them by application-specific means, or by ensuring that the protocol stack used to access the directory provides protection, for example by including the OSI NLSP)
  - — threats arising due to masquerade of a user of the XDS API or to a masquerade to a user of an implementation of the XDS API (these should be addressed by other parts of the environment at logon time or when a process attempts to change its *uid*).
- Because the directory service can provide access to security information, consideration should be given to enhancing the referenced **XDS** CAE Specification to cover security mechanisms for the other threats.
- These enhancements would enable security-aware applications to control these security mechanisms via the XDS API.

- The most appropriate means of control for other security mechanisms that are not currently visible through the API would be a common security mechanisms control API.
- For most applications, threats of denial of service are less serious than other threats.
- Because the directory service can provide access to security information, it should be considered whether the other enhancements should be made mandatory, even though existing implementations must then be modified if they are to continue to conform.

C.1.3 Impact on Other Specifications

If control of security mechanisms through systems management facilities is to be provided, there will be an impact on systems management specifications.

C.2 Overview of Security Facilities Provided

C.2.1 Security Goals

Implementations of the XDS should provide applications with secure access to a directory service.

C.2.2 Security Framework

See the referenced **X/Open Distributed Security Framework**, the body of this technical study and X.509 in the referenced **X.500** series CCITT recommendations.

C.2.3 Security Functionality and Services

All the X/Open security classes defined in the referenced **Secure Systems Procurement** X/Open Guide are potentially applicable to an implementation of the XDS and the applications that use it. As the XDS API supports the implementation of distributed applications and systems, the services that are grouped under the X-DIST security class are likely to be of particular relevance to XDS applications.

C.2.4 Security Information

Use of the XDS API in conjunction with the Directory Access Protocol (DAP) defined in the referenced **X.500** series CCITT recommendations is described in the referenced **XDS** CAE Specification.

Use of the XDS API in conjunction with the DAP defined in the referenced **X.500** series CCITT recommendations or the Domain Name Service (DNS) protocol defined in the referenced **DNS** RFC is described in the referenced **DCE Directory** Preliminary Specification.

With both of these protocols, the user data conveyed by the protocol could contain security information used by applications or by system components. This is particularly likely to be the case with user data conveyed by the DAP.

The DAP makes explicit provision for conveying the following types of security information:

- identification
- authentication
- integrity
- digital signatures.

While the protocol does not explicitly convey access type information, it provides separate constructs for read and modify operations.

The DNS protocol does not make explicit provision for conveying security information.

C.2.5 Standards

The security services and mechanisms for open systems interworking are defined in the referenced **OSI Security Architecture** standard.

The directory authentication framework is defined in X.509 in the referenced **X.500** series CCITT recommendations.

C.2.6 Emerging Standards

None applicable.

C.3 Security Specification

At present, the referenced **XDS** CAE Specification provides practically no support for security mechanisms.[9]

The following are specific conclusions in the light of the policies stated in Section C.1.2 on page 67:

1. Support for the following security mechanisms, as permitted by the security information conveyed by the DAP, could be provided by enhancements to the referenced **XDS** CAE Specification:

   - authentication exchange
   - data integrity
   - digital signature
   - notarisation.

   The DAP does not provide for encryption of user data. While this is not a problem when the user data contains public keys, as envisaged by X.509, it may be a problem if the directory is used to store security information for other security schemes which may, for example, use secret keys. The XDS specification could draw the reader's attention to this fact.

   The XDS specification could also draw the reader's attention to the fact that protocols other than the DAP (for example, the DNS protocol) may not support the security mechanisms listed above.

   Support for the security mechanisms listed above could be added to the XDS specification by defining new attributes for the **Session** and **Context** classes. (It should not be necessary to define new functions or to define new arguments for existing functions.)

   These issues should be considered by the X/Open XNET and Security working groups. These groups should first establish whether there is a requirement to provide support for these security mechanisms in the XDS. Any work on enhancing the XDS specification should take into account the extensions of the XDS that are defined in the referenced **DCE Directory** Preliminary Specification.

2. Support for the audit trail security mechanisms and for alarm generation following event detection could be specified in the XDS. However, it would be preferable not to do this. Rather, X/Open should make it possible for component definitions to require audit trail and alarm generation to be provided in conjunction with the XDS.

   The events recorded for audit would be normal protocol events (connections, disconnections etc.) and API calls, plus attempted security violations (calls resulting in errors of class **Security-Error**).

9. It does define a package called the Strong Authentication Package (SAP), but this is to enable the application to access security information that is stored in the directory, not to provide secure access to that information or to other information that is stored in the directory.

   An enhanced version of the XDS is used as the API to the DCE global directory service. For this purpose, some extensions of the XDS are defined in the referenced **DCE Directory** Preliminary Specification. They enable an application to provide a password for authentication purposes.

Attempted security violations would result in generation of alarms.

The interworking and systems management experts within the X/Open organisation should liaise to discuss a security API that would provide for control of the audit trail and alarm generation security mechanisms and that would be applicable to components that support the XDS.

3. There are security threats associated with memory allocation for private objects and service-generated public objects. To counter these threats, an implementation should ensure that:

   - other processes are prevented from accessing the memory that is used to store private objects and service-generated public objects used by a process
   - the contents of memory that has been allocated and returned to store are erased before the memory is re-used
   - there is a limit on the proportion of available memory that can be allocated to a single user.

   A prospective purchaser of components supporting the XDS should be aware of the security threats associated with memory allocation and should ensure that the implementation purchased provides the appropriate degree of security.

4. Authentication of the identity of the principal using the XDS is beyond the scope of the XDS specification, but the XDS specification could draw the reader's attention to the fact that it is important that the application, or other system components, ensure that this identity is authenticated. The X/Open XNET and Security working groups should consider whether it is desirable for the XDS specification to point this out.

The relevant interworking and security areas of expertise within the X/Open organisation have to decide:

- the options that would be appropriate

  and

- the mechanisms that should be required for each option.

Glossary

access control
The ability to grant or deny access to a resource.

API
Applications Programming Interface.

audit trail
A record of all the significant events that take place in a transaction or a series of operations.

authentication
The exchange of security information in order to verify the claimed identity of a communications partner. In the context of security, it is used in particular to counter attempts to masquerade as an authorised user in order to establish new connections or associations.

CCITT
Consultative Committee of International Telegraph and Telephone: an international committee whose membership is composed of government postal, telephone and telegraph agencies (PTTs).

component
In X/Open, the smallest unit that can be awarded an X/Open brand.

component definition
In X/Open, the definition which specifies the smallest elements of functionality that may separately be awarded an X/Open brand under the XPG4 branding programme.

data integrity check
A check to determine whether data has been modified.

digital signature
A piece of information which enables its origin to be verified.

encypherment
See encryption.

encryption
Encoding of information by transforming it in a way that is known only by its intended recipient(s), with the intent that this encoding can be reversed only by the intended recipient(s).

Internet
The cooperative virtual network that uses the TCP/IP protocol and includes the ARPANET, MILNET and NSFnet. It provides universal connectivity and reaches many universities, government, military and commercial establishments.

IPng
Internet Protocol next generation: a common term used to refer to the IP development activity due to be completed by the end of 1994.

IPv6
Internet Protocol version 6. This is the same as IPng.

ISO
International Standards Organization.

KDS
Key Distribution Service: in DCE RPC, this provides certificated authentication and key

distribution.

NLSP
OSI Network Layer Security Protocol

notarisation
Certification by a third party that a piece of information (such as a digital signature) is genuine.

PS
Privilege Service: in DCE RPC, this provides certification of attribute information.

RPC
Remote Procedure Call.

security domain
A part of an operational system that forms a unit for security purposes, and to which a set of countermeasures is applied against threats.

security policy
the security policy of a user enterprise. This must be distinguished from implementation aspects of security policy.

security authority
The authority that administers the security policy.

traffic padding
Generation of spurious communications or spurious data within communications, with the intent to provide protection against traffic analysis.

TLSP
OSI Transport Layer Security Protocol

X.400
The X.400 Message Handling Service (MHS) defined in the X.400 series CCITT recommendations. Also referred to as MOTIS.

X.500
The X.500 Directory Service, defined in the X.500 series CCITT recommendations.

XDCS
The X/Open Distributed Computing Support programme.

XPG
X/Open Portability Guide. Formerly the X/Open Portability Guide. XPG is the complete documentation for the process by which X/Open-compliant systems arrive at the marketplace, covering interoperability as well as portability.

Index