ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

STANDARD ECMA-138

SECURITY IN OPEN SYSTEMS

DATA ELEMENTS and SERVICE DEFINITIONS

December 1989

Free copies of this document are available from ECMA,
European Computer Manufacturers Association
114 Rue du Rhône - CH-1204 Geneva (Switzerland)

Phone: +41 22 735 36 34 Fax: +41 22 786 52 31

ECMA
EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

STANDARD ECMA-138

SECURITY IN OPEN SYSTEMS

DATA ELEMENTS and SERVICE DEFINITIONS

December 1989

Brief History

ECMA, ISO and CCITT are working on standards for distributed applications in an Open System environment. Examples are the OSI Reference Model, the work on Open Distributed Processing and the Framework for Distributed Office Applications.

Security is a major concern in information processing. The security aspects of interconnection have been addressed by ISO in the work on the OSI Reference Model (ISO 7498-2, Security Architecture). The purpose of this ECMA Standard is to provide Data Elements and Security Services for use in the Application Layer. This ECMA Standard unifies many views of security needs and of security functionality, including notions about end-systems security, therefore it allows a coherent approach needed to realise secure Open Systems. This ECMA Standard is based on the concepts in ECMA TR/46: Security in Open Systems - A Security Framework.

This ECMA Standard proposes a set of Security Services and Security Information for use in the Application Layer of OSI. Protocols for accessing the Services and a Transfer Syntax for the Security Information make this ECMA Standard applicable to distributed systems. In doing so this ECMA Standard makes extensive use of the concepts developed in ECMA TR/42, Framework for Distributed Office Applications as well as in ISO/OSI standards. However, other concepts such as the object model of processing used in the work of ECMA/TC32-TG2 on Open Distributed Processing, may also be used to describe the Security Services developed in this document.

This ECMA Standard emphasises the need for specification of the externally visible and verifiable characteristics needed for the communication of security related information. However, it avoids placing unnecessary constraints upon the internal design and implementation of information processing system that process and exchange security related information.

This ECMA Standard is based on the practical experience of ECMA member Companies worldwide and on the results of their active participation in the work of ISO and CCITT as well as in national standards bodies in Europe and the USA. It represents a pragmatic, widely based consensus.

This ECMA Standard is oriented towards urgent and well understood needs and supports rapid and effective standardisation. It is intended to be capable of extensions to cover future developments in technology and needs.

Adopted as an ECMA Standard by the General Assembly of 14th December 1989.

Table of Contents

1. INTRODUCTION

This ECMA Standard defines data elements and services for the support of a wide variety of security requirements in a multi-user, multi-vendor distributed system environment. The data elements and services developed in this standard are based on concepts defined in ECMA/TR46: "Security in Open Systems - A Security Framework". Readers unfamiliar with security and the Framework document are encouraged to read ECMA TR/46 if they wish to fully understand this text. This ECMA standard is intended for designers and developers of standards for productive applications in open distributed systems.

This ECMA standard is the second in a series of documents on Security in Open Systems planned by ECMA TC32/TG9. The first document, ECMA TR/46, describes the underlying requirements and concepts for security in open systems. These two documents provide a basis for future development on subjects such as: authentication, access control, attribute management, secure associations, interdomain interchange, and security audit.

1.1 Scope

This ECMA Standard defines a number of abstract security services for use in a distributed system. For each of the abstract security services identified this document gives a service definition. These service definitions are intended to be used in contexts such as:

- Open Systems Interconnection where they may be used as references for the development of security related protocols or elements of other protocols,
- Open Distributed Processing where they may be used as references for the development of secure systems architectures,
- Applications Standards developments (e.g. Distributed Office Applications) where they may be used to specify security functionality embedded in specific applications.

The security requirements of distributed applications that are specific to the nature of these applications (e.g. access controls to the objects owned by a given application such as a database manager) are addressed here only with regard to the interactions of such applications with the security services.

This ECMA standard is structured as follows:

- Clause 1 (this Clause) gives a general introduction, references, and definitions of terms.
- Clause 2 describes a model of security based on the abstract Security Facilities in ECMA TR/46. A set of security services and the requirements for security information are described.
- Clause 3 describes the most important piece of security information used in the application layer: the Security Attribute. Methods for passing attributes between security services, and between application processes, are suggested. A standard format for packaging together and sealing attributes for transmission is given.
- Clauses 4 to 9 outline the individual security services, their semantics, service interfaces, and management interfaces.
- Clause 10 shows how the components described in this ECMA Standard should be used in other standards for productive applications.
- Appendix A contains the full ASN.1 for the data elements defined in the main text. Appendix A is part of this standard.
- Appendices B to F contain additional descriptive material which may help the reader to interpret the standard in their own environment.

A reader interested in security for distributed systems should read the document as laid out. A reader who is looking for help on providing security in a new application standard should read clause 10, then refer back to the earlier clauses as required.

1.2 Field of Application

Generally, security refers to a complex of procedural, logical and physical measures aimed at prevention, detection and correction of certain kinds of misuse, together with the tools to install, operate and maintain these measures. For the purpose of this ECMA Standard "security" will refer to characteristics of data processing systems that give resistance to attack and misuse, intentional or otherwise. Other aspects of systems security such as reliability, availability and redundancy, are outside the scope of this ECMA Standard.

Given the above definition, security addresses not only attacks and threats originating externally, i.e. by persons not belonging to the organisation operating a given network or system, it also addresses internal attacks and threats coming from known persons. By providing guarantees of integrity and or confidentiality of information, secure systems may be used to perform business transactions in such a manner as not to expose their users to unacceptable liabilities.

More and more computers are linked together in systems that provide a wide variety of services to their users. This ECMA Standard defines a unified set of security data elements, and security services for such systems.

In this ECMA Standard a number of design decisions have been taken in which the security functionality required in a distributed system has been distributed amongst a set of services. The services and their interfaces have been carefully chosen to represent consensus and the best available experience. Specific design freedoms for subsequent system builders have been identified and protected in the development of this standard, in particular, the widest possible choice of security policy has been left to the system designer. Only those aspects of security requiring interworking conformance have been fully prescribed in this ECMA Standard.

The concepts of security policy, trust and security domains are fundamental to this ECMA Standard.

1.2.1 Security Policies

The security requirements of organisations differ with business goals, operating environment, etc. These differences in requirements are reflected in the security policy models developed for different types of organisation. This ECMA Standard is not concerned with the merits and drawbacks of specific policy models. Instead care has been taken in this ECMA Standard not to exclude specific policies.

Particular attention has been paid to the support of the following types of policy:

- those most naturally implemented by the use of Access Control Lists;
- label based policies with, or without, information flow control;
- those most naturally implemented by the use of Capabilities;
- policies in which the accessing subject's access privileges are compounded from more than one source (see Clark/Wilson) and other policies involving the security attributes of the route by which access is being requested;
- combinations of the above.

1.2.2 Trusted Systems

Without attempting to define or specify secure systems this ECMA Standard assumes that some parts of the real systems used to run the security servicessecurity service, as defined in this ECMA Standard, are trusted in some sense by the organisation using them. Such trust

may be based on certification by a third party or on internal accreditation of the system; establishment of trust is outside the scope of this ECMA Standard.

Assuming that trusted systems form the basic components of distributed systems leaves the problem of the insecure environment in which these systems operate and which they must use to communicate. Unless secure channels are available, the exchange of security control and management information must be protected against disclosure, replay and modification.

1.2.3 Security Domains

Security controls are the expression of a policy. Policy execution is performed and controlled by specialist Security Administrators collectively referred to as a Security Administration. The purview of a Security Administration may be referred to as a Security Domain. Within a Security Domain the Security Administration is responsible for the security of the installation, maintenance, and day to day running of the secure systems that make up the domain. Within a domain under one Security Administration, there may be more than one Administrator: separate Administrators with distinct identities may be used for different control aspects. For example subject administration and resource administration may be exercised by two distinct Administrators. Clause 1.5 contains a full definition of a security domain.

A sub-domain *inherits* the security policy of the super-domain and *refines* or *extends* it to produce a security policy of which the super-domain's policy is a subset. Ultimate control over the sub-domain is retained by the Administration of the super-domain, but direct control may be delegated. Although the super-domain defines specific rules only for the action types relevant to that domain's policy, there is a need to specify whether actions of other types are all permitted or all disallowed within the domain. Entities which are viewed as single elements in the super-domain may be redefined as multiple elements in the sub-domain, each of which may be subject to access rules pertaining to new action types.

1.2.4 Interdomain Propagation of Trust

Where organisations exchange security control information they may agree on recognising distinct Administrators, one associated with each domain. The mechanisms used in interdomain exchange are within the scope of this ECMA Standard. The translation of local authority into interchange authority is the purpose of the Interdomain Facility identified in the Security Framework. In this ECMA Standard the Interdomain Service models the functionality needed for interdomain exchange.

1.3 Conformance

In order to conform to this ECMA Standard the security data elements of an implementation shall conform to the specifications found in clause 3 and Annex A.

The description of the Security Services identified in this ECMA Standard is such that conformance of an implementation to these Services is not testable. However, these Services are part of an overall security architecture. To be in accordance with this architecture an implementation would use these Services in the manner described in this ECMA Standard and in ECMA TR/46.

Clause 10 of this ECMA Standard gives advice to application standards designers in their specifications of conformance statements relating security.

1.4 References

1.4.1 ISO and ECMA References

ECMA TR/46	Security in Open Systems - A Security Framework
ECMA-131	Referenced Data Transfer
ISO TR8509	OSI Service Conventions

ISO 7498	Open Systems Interconnection, Basic Reference Model
ISO 7498/2	Basic Reference Model, Security Architecture
ISO 7498/4	Basic Reference Model, Management Framework
ISO 8326	Open Systems Interconnection, Basic Connection Oriented Session Service Definition
ISO 8649/2	Open Systems Interconnection, Application Service Elements, Part 2 Association Control
ISO 8824/1-2	Specification of Abstract Syntax Notation One (ASN.1)
ISO 9594/1-8	The Directory (X.500)
ISO 9545.1	Open Systems Interconnection, Application Layer Structure
ISO 10031/1-2	Distributed Office Application Model
CCITT X.400	Message Handling System (Rec. X.400 to Rec. X.420)

NOTE 1
This ECMA standard outlines a number of services in terms of primitives and parameters. The notation used to specify the services is based on that in ISO TR8509 and ISO 8326. Realising that this notation is not completely appropriate for use in application layer standards, and that the specification of application layer services is currently being reviewed by ISO, we have limited the use of ISO TR8509 notation to the REQUEST and RESPONSE indications. In every case the INDICATION and CONFIRMATION indications correspond to the REQUEST and RESPONSE. Similarly, most of the primitives are confirmed and few of them have provider initiated actions. It is likely that protocols to implement these primitives will employ some form of Remote Operations, or similar underlying service.

1.4.2 Additional Sources of Information

- *Clark/Wilson* D. D. Clark & D. R. Wilson, A Comparison of Commercial and Military Computer Security Policies, Proc. 1987 IEEE Symposium on Security and Privacy, Oakland CA.
- *Report on the Invitational Workshop on Integrity Policy in Computer Information Systems (WIPCIS)*, NIST500-160 (US National Institute of Standards and Technology).

1.5 Definitions

For the purposes of this Standard the following definitions apply.

1.5.1 General Terminology

1.5.1.1 The following terms are used with meanings defined in ISO 7498:

Application layer
Application process
Application entity
Application service element
Presentation layer
Presentation connection
Protocol
Service definition

1.5.1.2 The following terms are used with meanings defined in ISO 8824/2:

Macro
Macro notation
Coordinated Universal Time

1.5.2 Specific Terminology

1.5.2.1 The following terms are used with meanings defined in ISO 7498/2:

Accountability
Access control
Audit
Channel
Confidentiality
Credentials
Data origin authentication
Identity based security policy
Integrity
Peer entity authentication
Rule based security policy
Security label
Trusted functionality

1.5.2.2 The following terms are used with the meaning defined in ECMA TR/46:

Access control list
Access control policy
Access context
Authentication policy
Authorisation policy
Control attributes
Privilege attributes
Referenced data transfer
Security facility
Security object
Security policy
Security subject
Trust

1.5.2.3 The following Terms are defined in this ECMA Standard:

1.5.2.3.1 Abstract Security Service

A set of security functions that together provide one or more of the security facilities defined in ECMA TR/46.

1.5.2.3.2 Application

Where the word "application" is used in this text it refers to a generic concept and should not be confused with the terms "application process" and "application entity".

1.5.2.3.3 Attribute Authority

An authority recognised in a security domain as a trusted source of attributes for entities within the domain. (See also Subject Authority).

1.5.2.3.4 Audit Information

Information recording security events in the system.

1.5.2.3.5 Authentication

The process by which the identity of an entity is established.

1.5.2.3.6 Authentication Authority

An authority recognised in a security domain as a source of certified identities. (See also Subject Authority).

1.5.2.3.7 **Authorisation**

The process by which an access control decision is made and enforced.

1.5.2.3.8 **Authority**

An entity recognised by some set of secure systems as a trusted source of security information.

1.5.2.3.9 **Capability**

A privilege attribute used as an identifier for a resource such that possession of the privilege attribute confers access rights for the resource.

1.5.2.3.10 **Certificate**

Security data sealed by an Authority. The certificate contains the security data and the seal.

1.5.2.3.11 **Certified Identity**

An identity in the form of an attribute in a certificate. The issuing authority will have authenticated the owner of the certificate.

1.5.2.3.12 **Control Attribute Package**

A collection of control attributes associated with a security object.

1.5.2.3.13 **Identity**

A unique piece of information which is recognised as denoting a particular entity within a security domain. The identity information is only unique within the domain.

1.5.2.3.14 **Independent Security Domains**

Two security domains are independent if and only if:

1. they are administered by different administrations, and:
2. no dependency relationship exists between the security domains.

1.5.2.3.15 **Interdomain Authority**

An authority recognised by two or more Security Administrations as a trusted source of security information used between the respective Security Domains.

1.5.2.3.16 **Principal**

An initiator that is capable of initiating interactions on objects, and which is not acting on behalf of, or by proxy, of another object. A Principal can be either a human user or an active object.

1.5.2.3.17 **Privilege Attribute Certificate**

A certificate containing Privilege Attributes.

1.5.2.3.18 **Proxy (1)**

An entity in a system that acts on behalf of another entity in invoking some operation.

1.5.2.3.19 **Proxy (2)**

Privileges that allow an entity to act on behalf of another entity.

1.5.2.3.20 **Resource Authority**

An authority recognised in a Security Domain as a trusted source of security information which relates to resources (security object).

1.5.2.3.21 **Seal**

A checksum, which may be cryptographic, computed over some data to provide integrity for that data.

1.5.2.3.22 Security Administration

A human authority which establishes a security policy and identifies the entities to which the policy applies.

1.5.2.3.23 Security Attribute

A security attribute is a piece of security information which is associated with an entity in a distributed system.

1.5.2.3.24 Security Domain

A set of entities that is subject to a given security policy and a single security administration.

1.5.2.3.25 Security Policy

A set of rules which define and constrain the types of security-relevant activities of entities.

1.5.2.3.26 Security Service

A set of operations designed to support some aspect of security in a distributed system.

1.5.2.3.27 Security Sub-domain

A proper subset of a security domain's entities which are subject to a security policy which possesses the following properties:

1. it includes all of the rules of the super-domain's policy where they apply;
2. additional rules are defined for the sub-domain policy action types where permitted by the policy of the super-domain;
3. it is administered by an Administration authorised by the super-domain's Administration.

1.5.2.3.28 Subject Authority

An authority recognised in a Security Domain as a trusted source of security information concerning security subjects (human beings and Applications).

1.6 Acronyms and Abbreviations

ACL	Access Control List
ACSE	Association Control Service Element
AE	Application-entity
ASE	Application-service-element
CAP	Control Attribute Package
DESD	Data Elements and Service Definitions *(this Standard)*
OSI	Open Systems Interconnection
PAC	Privilege Attribute Certificate
PSAP	Presentation Service Access Point
ROSE	Remote Operations Service Element
SACF	Single Association Control Function
SMAE	Security Management Application Entity
SSA	Supportive Security Application
TSP	Target Security Parameters

2. SECURITY SERVICES MODEL

This clause develops a reference model for the description and specification of security in open distributed systems. The model is based on the Security Facilities identified in ECMA TR/46. It is not implementation specific and does not constrain the implementation to specific security policies.

2.1 Modelling Terminology

This model is developed in the object oriented modelling paradigm. This object based approach enables the model to cover a number of different sets of terminology and ways of looking at distributed systems, as well as providing a better basis for future development.

The ECMA TR/46 security model was developed in terms of *security subjects* and *security objects*; where security subjects were the initiators of some access to a security object. Security objects need protection from security subjects and security subjects need privileges to access a security object. The terminology of security subject and security object may be confusing and contradictory to the target audience of this standard. Consequently we have introduced the terms *initiator* and *target* where they may be confused with "subject" and "object" used in the security world. In the object model of this clause, security subjects and security objects are both objects. When taking a snap-shot of a system at any time some objects will be acting as security subjects (or initiators of interaction) and some will be acting as security objects (or the targets of interaction). All sequences of interactions need to be started, the object that acts as the first initiator in a sequence is known as a principal, all other initiators in the sequence will be acting as proxies of the principal.

In an object oriented model, data is encapsulated inside an object, just like an abstract data type. So there is no need to model applications and data as separate kinds of entities in the model, they are both objects. This means that the model will concentrate on security between objects and assume that some objects represent data in the system and that some objects represent processing. An application may be considered as a processing object. In a system design an application would consist of a group of objects working together, with the same protection and privileges.

Object interaction is the ideal way to model the *client-server* mode of interaction. The client is the initiator object and the server is the target object. Of course, as in the client-server paradigm, objects that act as targets at one point in a computation can act as initiators at other points.

In summary, this clause uses the object oriented modelling paradigm to encompass the terminology and modelling techniques used in other fields to make the application of this security model more general, and therefore more powerful. To avoid confusion this text will use the terms *initiator* and *target* to describe the two parties in any object interaction.

2.1.1 Development of the Abstract Model

In ECMA TR/46 the Security Facilities were introduced as the means to describe functions and interactions required to model a secure system under a security policy. This clause introduces the more specific concepts of Security Information and Abstract Security Services. These will serve as a basis for the specification of security functionality in Application Standards. The Abstract Security Services and Security Information are derived from the Security Facilities in ECMA TR/46 and the model described in this clause is a refinement of the model in ECMA TR/46. The abstract security services may also be used to design a secure environment for unsecure applications.

In a computerised system the two entities that have to be taken into account when considering security are: the human users and the objects that are being used to model the activity within the system. This applies to any system, whether distributed or not. These components are shown in figure 1, with their potential interactions.

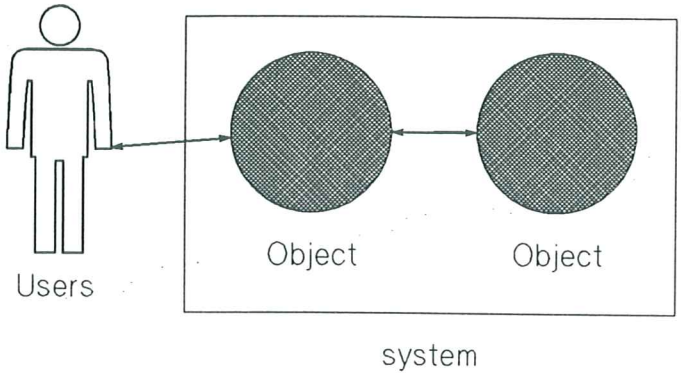

Figure 1 - Basic Model Components of a System

The purpose of a security model is to show how these two given components can be protected and their interaction controlled. The security services mediate between the human user and any objects in the system, they also mediate all interaction between objects themselves.

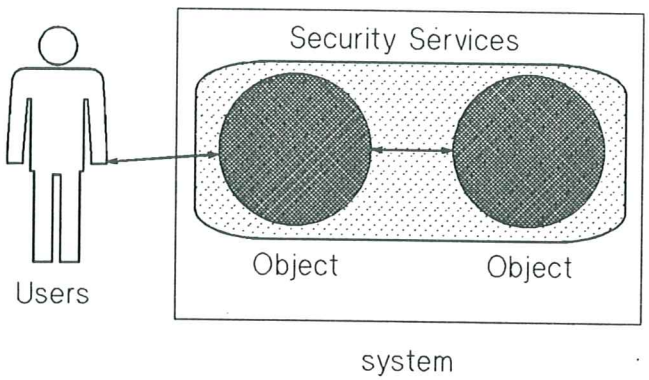

Figure 2 - Model Components and Relation to Security Services

The placing of the security services is shown in figure 2. Note that object interaction is independent of end-system boundaries, consequently this security model applies equally well to a single computer system as to a large distributed system.

The purpose of the Abstract Security Services is, in combination, to support control over the system according to some security policy. Three areas of control are supported by the security services in this model:

- Control by objects over access to their internal state, this means objects themselves can apply control to different aspects of their internal operation in a manner which is independent of the supporting environment. To do this the security services provide the object with the appropriate information and support in making and carrying out decisions.
- Control by the infrastructure over access to objects. In an object system the existence of objects and interaction between them is supported by an infrastructure. A part of that infrastructure is concerned with security and implements the control on object interaction. This infrastructure is provided with the necessary information and decision making support by the security services to exercise its control functions. This can be viewed as recursion of the model in which a single object is decomposed into component objects; the relationship between these component objects and their users is controlled by security services in the same way as the decomposed object.

- Control by the system over human user access. The model in this clause is based on the concept of a human user accessing a system. The security services support this system level access and provide information to the rest of the system on a particular user's identity and privileges. The security services provide the means to check users accessing the system and to allocate only those privileges that each user is entitled to under the security policy.

From these three aspects of control it can be seen that security can be applied to any level of granularity required. The basis under which control is exercised throughout the system is the use of security information.

Three types of security information can be identified:

Identification and Authentication Information.
This enables objects to identify and to authenticate human users and other objects within the system.

Control Information.
This information will specify, from a security point of view, what any entity may or may not do. The semantics of the security control information will be governed by the requirements of an individual security policy.

Security Monitor Information.
This information is used between the individual security services to audit and control the security of the distributed system as a whole.

The requirements for standardisation of Security Information are developed in this clause and a rigorous standard is given in clause 3. Service definitions for the Abstract Security Services are developed in this clause to the level of functional requirements for explicit operation and are further developed into outline service specifications in clauses 4 to 9.

2.2 Classes of Security Service

Security information is the cornerstone of the security model. Security services provide, process, and utilise security information. Objects need to obtain security information, store the information, and then use it to allow interaction with their environment. There are two classes of security services that are directly and actively involved with implementing security policy: the **security information class**, and the **security control class**. The security information class of service provides and processes security information, the security control class of service utilises security information.

It is necessary for the security in a system to be maintained in a consistent manner. For this purpose the security services need to be policed internally to detect attempted and actual violations of the integrity of the system and its information. A third class of security service, **security monitoring**, provides for the collection and processing of information relating to the operation of security functions.

2.2.1 Security Information Providing Class

There are three kinds of security information providing services: **authentication**, **security attribute**, and **interdomain**. These provide and process the two types of security information: authentication information related to the identities against which objects are accountable for their actions and Security Attributes used in the control of security functions e.g. access control. These three are the only types of service that provide trusted security information. Any such service represents one or more Administrators for issuing security information.

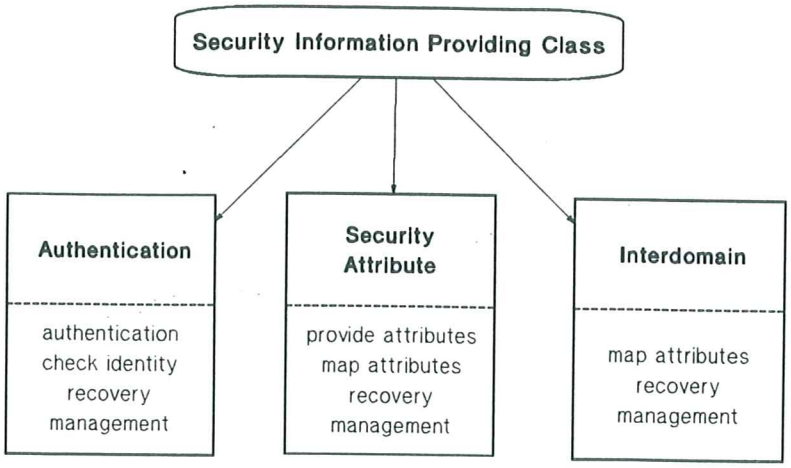

Figure 3 - Security Information Class Services and Operations

2.2.1.1 Authentication Service

There are two types of active entities in a distributed system: human users and objects. Both require authentication before they are allowed access to other objects. Objects may be required to be authenticated before they are accessed.

2.2.1.1.1 Human User Authentication

An Authentication Service authenticating human users accepts credentials as input and provides authenticated identities, if appropriate, in return. The proof of authentication is a certified identity issued by the service in its role as an authentication authority. An Authentication Service also includes a checking function that allows other security services, or objects, to request that identity certificates be checked.

An Authentication Service contains the following Security Facilities:

- the Authentication Facility which models the basic functionality of the Service,
- the Authorisation Facility which models the access control function needed for the management of the Service,
- the Recovery Facility which models the recovery behaviour of the Service

2.2.1.1.2 Peer Entity Authentication

See 2.2.2.2 Secure Association Service.

2.2.1.1.3 Object Authentication

Objects may act as two types of initiator: as an autonomous initiator (principal) or as an initiator acting on behalf of another object (proxy). In either case they must be capable of being authenticated and their access rights must be defined. This ECMA Standard assumes that such objects will have associated with them the security information necessary for their authentication and/or for access control decision making. The process for attaching this information is outside the scope of this ECMA Standard, only the syntax and, where appropriate the semantics, of this information is defined. This information may take the form of access rights that can be used directly in access control decisions. Where this information takes the form of credentials, an Authentication Service must be used to verify these and to issue the required access control information.

2.2.1.2 Security Attribute Service

A security attribute is a piece of security information which is associated with an entity in a distributed system. If the attribute is associated with an active entity (for example: a human user or initiator object) then it is known as a Privilege Attribute that indicates the privileges of the entity. If an attribute is associated with a passive entity (for example: a target object)

then the attribute is known as a Control Attribute, and it will control the access to that entity. A Security Attribute Service will provide attributes for entities on presentation of the entity's authenticated identity. Attributes may be sealed by the service, which acts as an attribute authority, so that they cannot be used in association with any other entity. As security information, Privilege Attributes may be passed between objects as a mechanism for conferring privileges on other objects. A Security Attribute Service will also map attributes where required, e.g. between distributed system-wide attributes and the particular attributes used within an end-system when the attribute authority remains the same. Further discussion on the nature of attributes is given in clause 3.

A Security Attribute Service contains the following Security Facilities:

- the Attribute Management Facility which models the basic functionality of the Service,
- the Authorisation Facility which models both the rule mechanisms needed in attribute mapping as well as the access control function needed for the management of the Service,
- the Recovery Facility which models the recovery functions of the Service,

2.2.1.3 Interdomain Service

A security policy applies to a single security domain. Such a domain delimits the responsibility as well as the recognition of its security authorities. An Interdomain Service provides the means for mapping security authorities and Security Attributes between domains. In doing so, it provides for the sealing of identities and attributes by a Security Authority recognised in the target domain. Implicit in this function is a control aspect that governs decisions about allowing exit from and entry into the domain. The target domain may be a sub-domain, or an independent domain, or a superior domain.

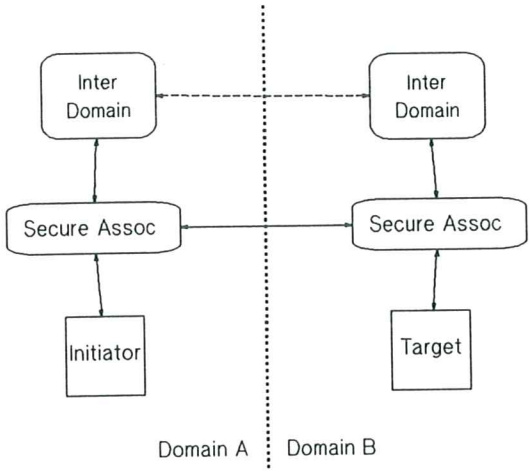

Figure 4 - Relationship between Interdomain Service and Secure Association Service

The model for an Interdomain Service is shown in figure 4. Associations over domain boundaries (between objects in different domains) are made directly via a Secure Association Service for each object. The initiator Secure Association Service component may use an Interdomain Service to map any association level security information before sending them to the target Secure Association Service component. This mapping may involve an authorisation decision. A receiving Secure Association Service component may send any received security information to an Interdomain Service for mapping and/or checking. Once an association is open, an object may use an Interdomain Service to map any application specific security information to be used during the association.

To facilitate mapping and proper recognition of remote Interdomain Administrations, interaction may take place between Interdomain Services of communicating domains. This is shown as a dotted line in figure 4, indicating that it does not take place as a direct result of the initiator-target interaction, but is a necessary prerequisite.

There is no explicit Interdomain control between Secure Association Service components. Control may be applied by the Authorisation Service invoked by Secure Association Service component, or by the refusal of the Interdomain Service to map, or verify any security information passed to it (see appendix D).

In principle, the security attribute mapping may be performed on exit from as well as on entry into a domain. The actual mappings performed are determined by policy. For efficiency reasons, Domain Administrators may agree between them to use a common syntax specific to their domains. Alternatively, a universally agreed attribute syntax may be used that is acceptable to any Domain Administration. Clause 3.7 provides such an attribute syntax. Using this "interchange" attribute syntax avoids NxN syntax translations in a multi-domain environment.

An Interdomain Service contains the following Security Facilities:

- the Interdomain Facility which models the basic functionality of the Service,
- the Authorisation Facility which models the access control function needed for controlling inter-domain activity and for the management of the Service,
- the Recovery Facility which models the recovery behaviour of the Service,

2.2.2 Security Control Class

In secure systems, controls must be applied at various points of interaction as shown in figure 1. These controls have to do with information collection and with information checking (as in access authorisation). Such information processing provides the basic operation of security control.

The basic kinds of security control services are: **Secure Association Services** to control the security aspects of interactions between objects, and **Authorisation Services** to provide access control decisions. In addition there is the **Subject Sponsor** security functionality to support the interaction with human users,

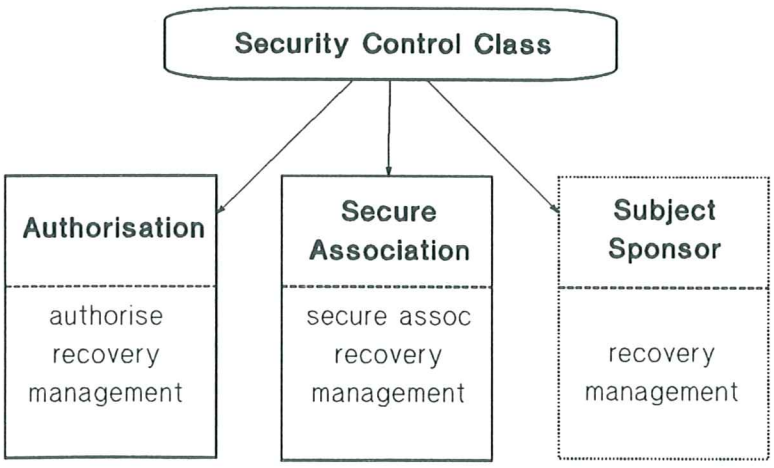

Figure 5 - Security Control Services and Operations

An access control policy will be implemented by two mechanisms: the allocation of Security Attributes to users and objects, and Authorisation. The allocation of Security Attributes to users and objects in the distributed system will be governed by long term, or essentially static, policy requirements. The attribute allocation algorithms will be subject to change by a Security Administrator in response to external events. Short term, or dynamic, control is provided at the point of authorisation, where the attributes are actually used. This control will take into account the total context of each request and will thus provide fine grain security policy management.

2.2.2.1 Authorisation Service

Access to entities subject to a security policy is controlled by an Authorisation Service. The primary inputs for the decision process are the initiator and target Security Attributes. Secondary inputs may be various factors relating to access context, such as time of day. These are outside the scope of this ECMA Standard. The access control process has three stages: gathering information, taking a decision, and enforcing the decision. The decision making component is provided by an Authorisation Service, the enforcement of the decision is the responsibility of the security infrastructure (see 2.3) and is outside the scope of this ECMA standard.

An Authorisation Service contains the following Security Facilities:

- the Authorisation Facility which models the basic functionality of the Service as well as the access control needed for the management of the Service,
- the Recovery Facility which models the recovery behaviour of the Service,

2.2.2.2 Secure Association Service

A special type of functionality that is necessary to support interactions within a distributed system is that of making an association. The Security Attributes of the target and initiator, and the access context, are used to authorise the association. A Secure Association Service is also responsible for supporting and/or enforcing the use of additional communications security functions on the association.

Peer Entity Authentication services are provided indirectly by a Secure Association Service on request. Details of methods of peer entity authentication and the associated protocols are excluded from the scope of this ECMA standard, though in practical implementations it is expected that Secure Association Services will use OSI security services in conjunction with a Security Infrastructure.

A Secure Association Service has a component in each end-system that provides the Secure Association Service to an object. The functionality is provided by combining a number of communications security services as required by the security policy. A Secure Association Service contains the following Security Facilities:

- the Association Management Facility which models the basic functionality of the Service,
- the Authorisation Facility which models the access control function needed for authorisation of the associations and the management of the Service,
- the Recovery Facility which models the recovery behaviour of the Service.

2.2.2.3 Subject Sponsor

ECMA TR/46 identified the Subject Sponsor as the trusted facility that provides the interface to the human user. A Subject Sponsor is primarily intended to interface to human users, but the Subject Sponsor may also be used to sponsor any active entity into a system. The primary roles of a Subject Sponsor are:

- to arrange for the authentication of the subjects it supports, by causing the exchange of credentials between the subject and an Authentication Service;
- to arrange an authenticated subject's connections to the objects it requires, ensuring that the appropriate access privileges are presented when required.

The Subject Sponsor has no interface operations that may be invoked by an other Security Service or object (apart from management and recovery which are generic); The Subject Sponsor is a user of the other Security Services. A Subject Sponsor contains the following Security Facilities:

- the Subject Sponsor Facility which models the basic functionality,
- the Authorisation Facility which models the access control function needed for management,
- the Recovery Facility which models the recovery behaviour,

2.2.3 Security Monitor Services

Security Monitor Services provide the means to verify the integrity of the security system and the means to correct its behaviour in response to observed or expected deviations. The integrity of the security system is maintained by the use of two security services: audit and recovery.

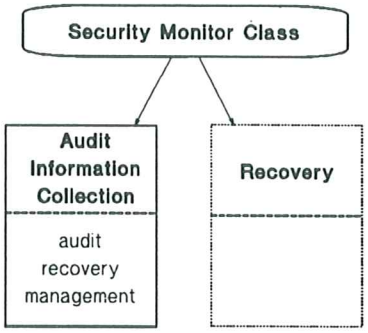

Figure 6 - Security Monitor Services and Operations

2.2.3.1 Security Recovery

When the integrity of the security system, or the distributed system as a whole is threatened, or actually violated, then recovery actions must be taken. This requires the use of Security Recovery as an integral component of each of the individual security services.

A recovery operation is defined for each of the actual security services in clauses 4 to 9, it is not by itself a service. The recovery operation can be invoked by other security services as required. A common format for the abstract recovery function is used to assist in its implementation. With each service, a list of possible recovery operations are given where appropriate.

2.2.3.2 Security Audit Information Collection Service

The Security Audit facility of ECMA TR/46 includes the collection and analysis of audit information. The model of security services in this clause recognises that analysis of security audit will be determined by the security policy of the domain. Consequently the model only supports the collection of security audit information as a standard service. An Audit Information Collection Service accepts information from the other security services. The nature and type of information to be collected, the analysis of this information, and the

resulting actions will all be determined by security policy. Objects will be able to use the security audit service to log their own security information as required.

A Security Audit Service contains the following Security Facilities:

- the Security Audit Facility which models the basic functionality of the Service,
- the Authorisation Facility which models the access control function needed for authorising the acceptance of audit records and the management of the Service,
- the Recovery Facility which models the recovery behaviour of the Security Audit Service itself.

2.2.4 Relationships in the Security Model

The security services described above to support security in a distributed system are closely related to the Security Facilities described in ECMA TR/46. Table 1 summarises the relationship between the services and the facilities of ECMA TR/46. This table shows that the security services model in this standard utilises the security framework in ECMA TR/46.

SERVICE / FACILITY	Authen-tication	Security Attribute	Secure Assoc	Inter-Domain	Author-isation	Audit
Authentication	S					
Attribute Management		S				
Association Management			S			
Inter Domain				S		
Authorisation	C	C	C	C	S	C
Audit	O	O	O	O	O	S
Recovery	C	C	C	C	C	C
Cryptographic Support	O	O	O	O	O	O

S = Service implements the facility
C = The service contains the facility
O = The service optionally contains the facility

Table 1 - Relationship between Security Facilities and Security Services

2.2.5 Other Security Services

The model of Security Services developed in this clause provides the basic set of services which are required to support a general level of security in an open distributed system. Other security services may be required to support specific mechanisms and security policy requirements. Examples of such services include:

- Notary Service - to support non-repudiation in object interaction.
- Key Management Service - to support the generation and distribution of cryptographic keys.
- Data Flow ControlService - to control the movement of data between objects.
- Labelling Service - to attach and verify labels with objects (see 3.6).

These services are outside the scope of this ECMA standard.

2.3 Security Infrastructure

In an end-system which is required to provide security to support a particular security policy it is necessary to have some trusted functionality - the security infrastructure. The trusted component

will be required to exercise control over the un-trusted parts of the system. In this security model, objects are only trusted to protect their internal state and data. The trusted control functionality needs to be implemented in some system dependent way, usually within an operating system with hardware assistance. The *security infrastructure* embodies this trusted functionality of control.

The security infrastructure interfaces to the security services through the appropriate protocols to obtain and check security information. In this way it is a user of the services. It will impose the authorisation decisions, having first obtained the decision from the authorisation service. In some cases the authorisation service may be implemented on the end-system and be part of the security infrastructure. The infrastructure will map distributed system wide attributes onto local capabilities and access control list parameters before applying access control. The control aspects of the Association Control Service will be implemented in the security infrastructure, other aspects of the communications security will be provided by specific ASEs from the OSI standards. The security infrastructure will support the recovery and audit functions of any security services on the end-system. Where the end-system is supporting a security application, such as authentication, then the security infrastructure will still be necessary. It will be the subject sponsor for the security application, just as it sponsors all applications. The security infrastructure will be responsible for the low level key management needed to support the other security functions.

2.4 Trust Relationships in the Model

The trust relationships explored are those *between* services relating to the correct use of security information and its authenticity.

The prime objective of this security model is to lead to the ultimate definition of standard protocols that enable secure interworking of objects in a distributed system. The achievement of security depends on the validity of certain assumptions about the ways in which different components of the model depend on each other to act responsibly and correctly. This clause lists these assumptions under two different categories: assumptions relating to the environment in which the model components exist (i.e. the infrastructure supporting the existence of the model entities themselves), and assumptions relative to the different degrees and kinds of trust that can be put in the model components.

2.4.1 Trust in the Security Infrastructure

This security model assumes that there is an underlying security infrastructure in the actual distributed system. This infrastructure is typically provided by operating systems and hardware on individual end-systems.

The infrastructure will provide trusted functionality to meet the requirements of this model. This functionality will be provided in an end-system dependent, and often proprietary, manner. The functionality required of the security infrastructure is:

- Protect the security services and objects against external corruption.
- Provide the OSI communications security services (for example peer entity authentication) requested by a Secure Association Service when forming associations, or indicate that it is unable to do so.
- Provide the necessary initial security information to security services and objects. For example cryptographic master keys and identities for security service applications.
- Enforce use of an appropriate Authorisation Service for each access request.
- Direct all initial human user contacts with the system to a Subject Sponsor and secure this link according to the local security policy.

2.4.2 Trust in the Security Model

The primary mechanism for the propagation of security in the Services Model is the use of Security Information. Security Information is generated by the Services representing the Security Administration of a domain: the Authentication Service, the Security Attributes Service and the Interdomain Service. Specific trust relationships in the Security Services model can be identified as follows:

- The security administration for the domain trusts all of the components of the system to generate appropriate audit information and function properly in all respects listed below.
- All of the security services and objects trust the Audit Service to which they send audit records. The Audit Service is trusted not to lose, reveal or corrupt audit records.
- All security services and objects trust a Secure Association Service to: ask for the appropriate level of security, to connect to the service requested, not to retain or reveal any information under its control, and ensure that the association is authorised.
- A Secure Association Service trusts an Authorisation Service to supply correct access control decisions when asked.
- A Subject Sponsor is trusted by its subject to request connection to an Authentication Service and objects via an appropriately secure connection, to pass data transparently between the subject and other components of the system without modification, to issue security information only as requested by the subject and only to the objects requested, and to close down an association when required. The Security Administration for the domain trusts it to time out inactive subjects according to policy and rescind security information on request from an Authentication Service.
- Each Authorisation Service trusts one or more Authentication and Attribute services to provide correct attributes.
- An Authorisation Service is trusted to make the proper validation checks on security information and to take access control decisions according to policy.
- An Interdomain Service trusts one or more Attribute Services to provide correct attributes.
- An Attribute Service trusts an Authentication Service to correctly authenticate users and objects.

3. DATA ELEMENTS

This Clause specifies the security related Data Elements used in this ECMA standard.

These Data Elements comprise the following types of security information:

- Security Attributes used in access control decision making;
- certified packages of Security Attributes referred to as Privilege Attribute Certificates, that express dynamic access privileges or rights;

This ECMA standard defines the syntax of the above data elements. Appendix E describes the relationship between the data elements defined here and those defined by the Directory Standard [ISO 9594].

3.1 Security Attributes

ECMA TR/46 describes how access authorisation decisions are made in the context of Security Attributes known as Privilege Attributes and Control Attributes possessed respectively by an initiator and a target. It also describes how, in a distributed system, the privileges of more than one initiator may be involved in an access authorisation decision.

The nature of a distributed system demands that initiator-related access privileges be capable of being transferred via communications protocols. As other clauses of this document show, these privileges in the form of a Privilege Attribute Certificate (PAC) are transmitted in many security protocols, and also in other protocols as additional security parameters. They are therefore central to the security of a distributed system and are dealt with at length below.

3.2 Privilege Attribute Protection

Privilege Attributes are particularly susceptible to misuse because they are frequently under the control of software entities which cannot be trusted not to tamper with them. They also need to be guaranteed by an authority. They therefore require a variety of forms of protection as shown below:

- protection against undetected modification,
- protection against use by the wrong initiator with or without the collusion of the intended initiator,
- protection against use at the wrong target,
- protection against use outside stated constraints (in particular after the privileges have expired).

Privilege Attributes are also used by objects on behalf of other objects, and this gives rise to another requirement for a form of protection:

- protection against use by the right initiator for the wrong purpose.

3.2.1 Protection Against Undetected Modification

This is achieved by binding together all of the Privilege Attributes under a seal provided by the Authority which issued them. Such a bound collection, coupled with the other control data described below is known as a Privilege Attribute Certificate (PAC).

3.2.2 Protection Against Misuse

This ECMA Standard identifies a number of schemes for protecting the PAC during transfer between objects and any subsequent misuse. The definition of the certificate in this ECMA Standard includes the necessary items to support these schemes. Use of one or more of these schemes will depend on the local security policy requirements. Since these schemes are designed to protect the receiving Authorisation Service from illegitimate PACs, it is the responsibility of the Authorisation Service to check and respond to each PAC presented to it.

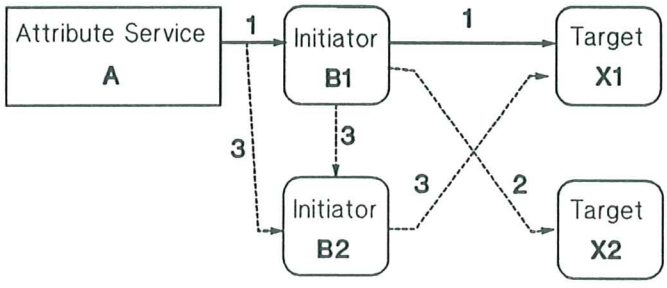

Figure 7 - Potential Misuses of Attribute Certificates in a Distributed System

3.2.2.1 Protection Using Qualifier Attributes

If Security Attribute Service A provides initiator B1 with a set of Privilege Attributes for use with the Authorisation function of target X1 (path 1 in Figure 7), it may require to ensure that B1 does not use them withtarget X2 (Path 2 in Figure 7) or that B2 cannot use them, having obtained them from A or B1 by some means (path 3). B1 can be

prevented from using the attributes with X2 if A binds X1's identity to the attributes. B2 can be prevented from using the privileges if B1's identity isalso bound to the attributes. Thus we have a tuple: [*B1-id,X1-id, privilege attributes*].

However, A may not know or care precisely which object will service B1's requests, but may require only that it possesses certain attributes itself (e.g. it belongs to the class "File Server"). The element *X1-id* of the tuple is therefore generalised to become any subset of the Control Attributes of the intended target viewed as an accessed object. The subset of an object's Control Attributes used in this way are called qualifier attributes. Similarly, the element *B1-id* is generalised in the same way so that B1 may legitimately pass on its privilege attributes for use by another object, but only to objects satisfying the given qualifier attribute conditions. These qualifier attributes can then be checked by the Authorisation Service of the target.

To use qualifier attributes to protect the PAC an Authorisation Service carrying out the check will need to correctly identify the actual entity passing or receiving the PAC. This requires peer entity authentication. The peer entity authentication facility of the Secure Association Service may be used when the target and initiator begin the activity. This protection scheme does not require any confidentiality services.

3.2.2.2 Protection Using a Validation Key

The validation key scheme provides protection by employing an additional item which is used in the calculation of the seal in the Privilege Attribute Certificate. This additional item would typically be an initialisation vector for the sealing checksum algorithm. The validation key is additional to any other key used in the sealing checksum algorithm. Thus this scheme is only appropriate where the chosen sealing checksum algorithm allows the use of both of these keys.

Whenever an authorisation service needs to check a PAC it will need to have access to the validation key. When the validation key is transferred between entities it must be protected against disclosure, possibly by selective field confidentiality. The confidentiality facility of the Secure Association Service may be used. This protection scheme does not require peer entity authentication when the PAC is transferred.

The validation key limits the use of the PAC to those who know its correct value. The validation key may be used in a challenge-response or similar exchange at the time of submission of an access request. Also, if the target is presented with the validation key via a separate route from that used for the PAC, then a stolen PAC cannot be used elsewhere. Protocols for these techniques are outside the scope of this standard.

3.2.2.3 Protection Using Communication Security Services

Privilege Attribute Certificates may be protected during transmission between end-systems by means of end-to-end OSI security services selected via the Secure Association Service from ISO 7498/2.

3.2.2.4 Protection Against Use Outside Time Constraints

The validity of a PAC can be limited to particular time periods if they are specified under the PAC seal.

3.2.2.5 Protection Against Use for the Wrong Purpose

An initiator can be constrained to act only with respect to a particular operation by using operation-specific Privilege Attributes in the PAC. A particular example of such a protection would be the RDT-reference associated with a Referenced Data Transfer [ECMA-131, ISO 10031/2] (see 10.1.3).

3.2.3 Compound Objects and Proxy Relationships

It is a common requirement in distributed systems for objects to request other objects to perform actions for them on their behalf. The intermediate objects are known as proxies. In some cases the proxy requires no additional authority from the initiating object to perform the action; in other cases additional authority is required. The combination of several initiators is known as a compound object. An example is shown in Figure 8. Object A (acting as a principal) wishes to access another object (X) which is protected by an access Authorisation Service. Object A is using object B as a proxy for doing this. Both A and B may possess Privilege Attributes that are relevant to the access, and the Authorisation function may require to be presented with the Privilege Attributes of both parties. For example if A is a human user asking Print Server B to read his file from File Server X, the File Server may require evidence both of initiator A's identity and, say, the security clearance of B (if the file is a company-secret file the file server may only release it to a Print Server that has company-secret clearance). This picture generalises to any number of objects, with the Authorisation Service of the final target object making an access decision in the light of the Privilege Attributes of each object in the invocation sequence. Notice that the compound object approach enables an Authorisation Service to be provided with the information necessary for it to support commercial Clark-Wilson style policies across distributed systems (see Clark/Wilson).

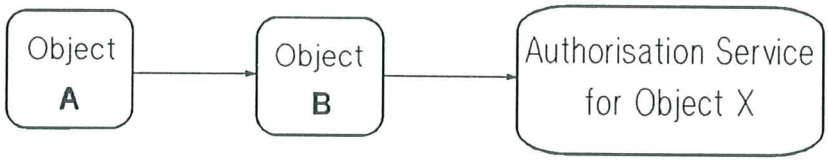

Figure 8 - Access by Compound Objects

3.2.4 Accountability

For audit purposes it must be possible to retrieve the identity of an initiator to which a set of attributes is first issued, and which is subsequently accountable for their use in accessing other objects. There are three ways of doing this; the first two arise when there is no requirement for anonymity:

1. An audit identifier field (auditIdentifier) may be present in the PAC; it contains the principal's own identity and is used for audit purposes.
2. When an initiator's identity is one of its access privileges (i.e. is being offered to an Authorisation Service in support of access requests) then that identity Privilege Attribute may also be used for audit purposes.
3. When anonymity is required, the audit identifier field in the PAC contains a reference number, unique for each principal, generated by the Attribute Service which issued the PAC. The identity of the initiator can be unknown to the target, but an audit manager who has access rights to the issuing Attribute Service's and target object's audit trails is able to achieve an active object's accountability by linking the active object's identity with the audit identifier.

3.2.5 Recovery

To enable a PAC to be revoked it is useful to have an identifier specifically for this purpose. The identifier may have a value which enables a number of PACs to be revoked with the same identifier, or range of identifiers. The recoveryIdentifier field is included in the PAC for this purpose.

3.3 Privilege Attribute Certificate Syntax

The full syntax of the Privilege Attribute Certificate is designed to meet three requirements:

- the minimum PAC consisting of a single attribute and a seal may be constructed easily with a minimum overhead,
- a compound structure including additional PACs and further attributes for use by compound objects and proxies where multiple objects or authorities are involved,
- support for the PAC protection schemes identified in this standard.

The syntax for the PAC follows (see Appendix A for the full definition)

```
-- this is the sealed unit which is sent down the wire
PrivilegeAttributeCertificate ::= SEQUENCE {
  SEQUENCE {
    -- identifies the version of the PAC syntax
    syntaxVersion INTEGER (version1 (1)),

    -- for a compound PAC:
      containedPACs          [0] SEQUENCE OF PrivilegeAttributeCertificate OPTIONAL,

    -- the actual privilege attributes:
      privilegeAttributes    [1] SEQUENCE OF SecurityAttribute,

    -- fields used to support PAC protection schemes
      initiatorQualifierAttributes [2] SEQUENCE OF SecurityAttribute OPTIONAL,
      targetQualifierAttributes    [3] SEQUENCE OF SecurityAttribute OPTIONAL,
      validationKeyIdentifier      [4] BIT STRING OPTIONAL,

  -- fields for additional protection in use
      creationTime           [5] UTCTime    OPTIONAL,
      validityTime           [6] TimePeriod OPTIONAL,
      pacIdentifier          [7] INTEGER    OPTIONAL,
      recoveryIdentifier     [8] INTEGER    OPTIONAL,

  -- Identity fields for use by intermediate handlers
      auditIdentifier        [9] Security-audit-identity OPTIONAL,
      chargingIdentifier     [10] Security-accounting-identity OPTIONAL,

  -- fields used for the certification:
      pacAuthority           [11] Authority},
-- seal on the end to assist length calculation
pacSeal     Seal }

Authority ::= SEQUENCE { -- to permit a variety of naming schemes
  authorityType   OBJECT IDENTIFIER,
  authorityValue  ANY DEFINED BY authorityType }

Seal ::= SEQUENCE {
  algorithmIdentifier  OBJECT IDENTIFIER,
  keyIdentifier        OCTET STRING,
  seal                 OCTET STRING }
```

```
TimePeriod :: = SEQUENCE OF SEQUENCE {
                    startTime [0] UTCTime OPTIONAL,
                    endTime [1] UTCTime OPTIONAL }
```

The containedPACs field contains any PACs already obtained by an initiator which will be required for the operation. PACs are embedded in this way when the Authorisation Service of the target object does not recognise the Attribute Authority of the service which created the PAC. An intermediate object, or an Interdomain Service, may cause the original PAC to be sealed inside another certificate; the authority on the outer certificate being acceptable to the target Authorisation Service. The use of compoundPAC is a matter for further study.

The privilegeAttributes are the attributes which provide information for an Authorisation Service to make an access control decision.

The targetQualifierAttributes and the initiatorQualifierAttributes are used if protection is provided by qualifierattributes (3.2.2.1).

The validationKeyIdentifier is included if the PAC is to be protected by the use of the validation key scheme (3.2.2.2), in which case this field contains the key identifier. It does not contain the validation key itself.

The creationTime and validityTime fields may be used to apply time related constraints on the use and validity of the PAC (3.2.2.4).

The pacIdentifier field will contain an identity which is unique to this PAC.

The recoveryIdentifier field is used to contain an identifier suitable for use with the recovery procedures required by the security policy (3.2.5).

The use of the auditIdentifier to support anonymity has already been discussed (3.2.4), it will be used whenever any activity authorised with this PAC is audited.

The chargingIdentifier field will contain an identifier suitable for allocating resource usage charges associated with activity authorised with this PAC.

The seal (3.2.1) is computed using all of the fields present in the PAC and, if the validationKeyIdentifier field is present, the value of the validation key (3.2.2.2).

3.4 Examples

Figure 9 shows a PAC for an object possessing two Privilege Attributes: a clearance of HIGH and an identity of TOM. It was object TOM which caused the PAC to be created. The PAC may only be issued from an initiator having an identity qualifier attribute of SPONSOR7 and may be used only with a target having a clearance attribute of HIGH.

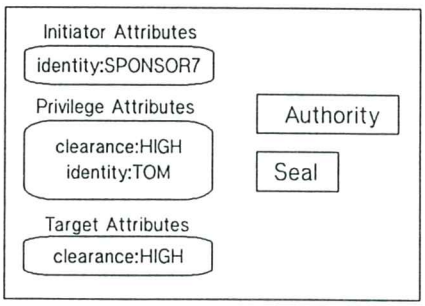

Figure 9 - First Example of a Privilege Attribute Certficate

Figure 10 shows a PAC for Subject JAN in operational role of CLERK for use from any initiator to any target.

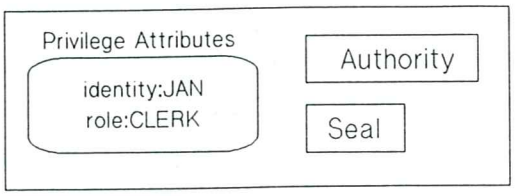

Figure 10 - Second Example of a Privilege Attribute Certficate

3.5 Control Attributes

Control Attributes are associated with target objects and are obtained from a source that can be trusted not to have tampered with them. Unless they are incorporated into a PAC where they are protected by the PAC's seal, they require only to be integrity protected during transmission when they are transferred with the object they are associated with. This standard defines a special data structure called "a labelled object" (see 3.6) which is used to bind control attributes to any object for the purpose of transfer and storage.

Where Control Attributes need to be transferred separately from the object to which they relate, for instance as a parameter to an Authorisation Service the Control Attribute Package is defined as:

```
-- Control Attribute Package for transferring Control Attributes
ControlAttributePackage ::= SEQUENCE OF SecurityAttribute
```

3.6 Labelled Objects

The labelled object uses similar mechanisms to that of the PAC to bind information data elements together and protect them against modification and misuse. The important difference with the labelled object is that the object itself is included in the certificate and that any attributes may be included in the label; thus making this a more general structure.

The syntax of the labelled object is:

```
-- Certificate encapsulating an object with its Control Attributes
LabelledObject ::= SEQUENCE {
    objectType          OBJECT IDENTIFIER,
    object              ANY DEFINED BY objectType,
    labelAttributes     SEQUENCE OF SecurityAttribute,
    labelSeal           Seal,
    labelAuthority      Authority }
```

3.7 Standard Attribute Types

Security Attributes are defined using the ATTRIBUTE macro of the ISO 9594 Directory Standard; this allows security attributes to be stored in the Directory. The semantics of each attribute is described below, and is not reflected in the ATTRIBUTE syntax. In particular the semantics for comparing attributes and for searching multi-valued attributes for particular values is included in the English text.

The attribute syntax for use with security attributes is the syntax used in the Directory Standard. The attributeValues syntax is additionally constrained to include an optional authority field (explained in the notes below). The resulting syntax of a security attribute is as follows:

```
SecurityAttribute ::= SEQUENCE {
    attributeType     OBJECT IDENTIFIER,
    attributeValues   SEQUENCE {
                          authority [0]          PolicyAuthority OPTIONAL,
                      SEQUENCE OF ANY }}-- DEFINED BY attributeType

    PolicyAuthority ::= OBJECT IDENTIFIER
```

NOTE 2
The values that are permitted for a particular attribute type are determined by the policy, in force in the security domain in which the attribute is to be used. These aspects of policy can be standardised by identifying commonly used sets of attribute values for a particular attribute type; one example of this might be the set of national hierarchic security markings for a particular country. There can be different distinct value sets defined for an attribute type according to the aspect of the access control policy it is being used to support. The authority field is used to define the particular value set that pertains for a use of an attribute type. The definition of particular value sets is out of scope of this document.

In a particular security domain there may be defined a security policy which demands that multiple sets of a given attribute type be supported, each set belonging to a different defining security authority, and being used to support a different aspect of the policy. This means that a distinction must be possible between attributes of the same type belonging to different sets. The authority field enables this distinction to be made. An example might be a system supporting two distinct label hierarchies, the first being a national government hierarchy, the second being a company seniority hierarchy. From an implementation point of view an application that supports multiple hierarchies is not concerned with what authorities they belong to but only that there are two distinguishable hierarchies.

Some attribute types are related in that they have complementary functions within the same security policy. For example different types are used to specify upper and lower bounds of the same kind of access privilege. By separating type from authority an Authorisation Service can more easily see the correspondences.

The following is not a complete list of security attributes, but is the basis for further development.

3.7.1 Hierarchic Ordered Access Control List Attribute

This attribute is an Access Control List (ACL) for use with protected objects that are hierarchically related. In this type of ACL the ordering of the entries is significant in that once an entry has been found identifying the requesting subject the access rights found are the ones used, no further search being undertaken.

NOTE 3
The syntax is for further study.

3.7.2 Hierarchic Unordered Access Control List Attribute

This attribute is an Access Control List (ACL) for use with protected objects that are hierarchically related. In this type of ACL the ordering of the entries is not significant; all entries will be searched to find the access requested if necessary.

NOTE 4
The syntax is for further study.

3.7.3 Ordered Access Control List Attribute

This is an ordered Access Control List for use with protected objects that are not hierarchically related.

NOTE 5
The syntax is for further study.

3.7.4 Unordered Access Control List Attribute

This is an unordered Access Control List for use with protected objects that are not hierarchically related.

NOTE 6
The syntax is for further study.

3.7.5 Accounting Identity Attribute

This attribute is used when accounting is required and the other attributes do not give the required granularity.

```
Security-accounting-identity ::= ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX IntegerOrString
        SINGLE VALUE
```

3.7.6 Audit Identity Attribute

This identity would be used when auditing is to be carried out on a different identity from that used for access control.

```
Security-audit-identity ::= ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX IntegerOrString
        SINGLE VALUE
```

3.7.7 Authentication Level Attribute

This attribute indicates which authentication level the entity has been authenticated to. It may also indicate the type of authentication that has been carried out.

```
Security-authentication-level ::= ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX INTEGER
        SINGLE VALUE
```

3.7.8 Capability Attribute

This is an attribute which grants access of the type accessType to the object(s) defined by objectDefiner.

```
Security-capability-type-1 ::= ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX CapabilityType1
        SINGLE VALUE

CapabilityType1 ::= SEQUENCE {
        objectDefiner ObjectDefiner,
        accessType AccessDefiner}

        ObjectDefiner ::= IntegerOrString
        AccessDefiner ::= IntegerOrString
```

3.7.9 Confidentiality Class

This attribute controls read access to an object in terms of categories of interest associated with the object as follows:

Let the set of categories of interest in the Confidentiality ClassAttribute associated with the initiator be I-Code; let the set in the Confidentiality Class Attribute associated with the target be T-Code. Read access is permitted (subject to any other constraints that might apply) only if I-Code includes T-Code.

Security-confidentiality-class :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX PrintableString
MULTI **VALUE**

3.7.10 Minimum Confidentiality Class Attribute

This attribute defines the minimum Confidentiality Class privilege that an initiator is permitted to operate with. For any particular initiator it appears paired with a Confidentiality Class attribute.

Security-confidentiality-class-minimum :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX PrintableString
SINGLE **VALUE**

3.7.11 Confidentiality Hierarchy Attribute

This attribute controls read access to an object in terms of a hierarchic measure of the level of confidentiality of the object as follows:

Read access is permitted (subject to any other constraints that might apply) only if the hierarchic level in the initiator's Privilege Attribute is at least as high as that in the target's Control Attribute.

Security-confidentiality-hierarchy :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX **INTEGER**
SINGLE **VALUE**

3.7.12 Minimum Confidentiality Hierarchy Attribute

This attribute defines the minimum hierarchic confidentiality level that an initiator is permitted to operate at. For any particular initiator it appears paired with a Confidentiality Hierarchy attribute.

Security-confidentiality-hierarchy-minimum :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX **INTEGER**
SINGLE **VALUE**

3.7.13 Entity Identity Attribute

This attribute is used to identify uniquely an active entity in the distributed system. This could be a human or an object.

Security-entity-identity :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX IntegerOrString
SINGLE **VALUE**

3.7.14 Group Attribute

This identity may be used to combine a number of individual identities, possibly for use with an access control list. This identifies one or more groups of which an entity is a member.

Security-group :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX IntegerOrString
MULTI **VALUE**

3.7.15 Integrity Class Attribute

This attribute controls write access to an object in terms of categories of interest associated with the object as follows:

Let the set of categories of interest in the Integrity Class Privilege Attribute associated with the initiator be I-Int; let the set in the Integrity Class Control Attribute associated with the target be T-Int. Write access is permitted (subject to any other constraints that might apply) only if I-Int includes T-Int.

Security-integrity-class :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX PrintableString
MULTI **VALUE**

3.7.16 Minimum Integrity Class Attribute

This attribute defines the minimum Integrity Class privilege that an initiator is permitted to operate with. For any particular initiator it appears paired with an Integrity Class attribute.

Security-integrity-class-minimum :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX PrintableString
MULTI **VALUE**

3.7.17 Integrity Hierarchy Attribute

This attribute controls write access to an object in terms of a hierarchic measure of the level of integrity of the object as follows:

Write access is permitted (subject to any other constraints that might apply) only if the hierarchic level in the initiator's Privilege Attribute is at least as high as that in the target's Control Attribute.

Security-integrity-hierarchy :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX **INTEGER**
SINGLE **VALUE**

3.7.18 Minimum Integrity Hierarchy Attribute

This attribute defines the minimum hierarchic integrity level that an initiator is permitted to operate at. For any particular initiator it appears paired with an Integrity Hierarchy attribute.

Security-integrity-hierarchy-minimum :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX **INTEGER**
SINGLE **VALUE**

3.7.19 Need-To-Know Attribute

This attribute controls read access to an object in terms of a initiator's need-to-know about any of the categories of interest associated with the target, as follows:

Let the set of need-to-know categories of interest in the Need-To-Know Privilege Attribute associated with the subject initiator be I-Need; let the set in the Need-To-Know Control Attribute associated with the target be T-Need. Read access is permitted (subject to any other constraints that might apply) only if the intersection of I-Need and T-Need is not empty.

Security-need-to-know :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX PrintableString
MULTI **VALUE**

3.7.20 Minimum Need-To-Know Attribute

This attribute defines the minimum need-to-know privileges that an initiator is permitted to operate with. For any particular initiator it appears paired with a Need-To-Know attribute.

Security-need-to-know-minimum :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX PrintableString
MULTI **VALUE**

3.7.21 Role Attribute

This attribute indicates the role that an entity is currently using in the distributed system.

Security-role :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX IntegerOrString
SINGLE **VALUE**

4. AUTHENTICATION SERVICE

Each service primitive has parameters for indicating the *result* of the service request (success, error, rejection, etc) and the reason for any *error* or *reject*. These parameters will be defined in detail in any protocols supporting the service primitive; they will not be described in any detail in this clause.

4.1 Operations Model

This service supports the authentication of entities on the basis of credentials presented during the authentication exchange. In this clause we use the word "entity" to represent "user or object".

To establish the set of primitives that an Authentication Service should support, the authentication status of an entity can be modelled in 3 states which are externally visible to an Authentication Service. Internally an Authentication Service will have many more states to control its operation. The three externally visible states are:

0. The entity is not authenticated, and authentication may begin. This is the initial state.
1. The entity is authenticated to use the system. A certified identity has been issued to the entity. Where applicable the system will have authenticated itself to the entity.
2. The entity authentication has been suspended, for whatever reason.

The purpose of the externally visible states is to identify the external primitives of an Authentication Service which can change the state of an entity. An Authentication Service provides the certified identity of the entity as the initial PAC that can be used to obtain other PACs. An Authentication Service is the authority for issuing certified identities, thus the certificate is sealed by the issuing Authentication Service.

Certified identity PACs may be subsequently presented to an authentication service to be checked for validity.

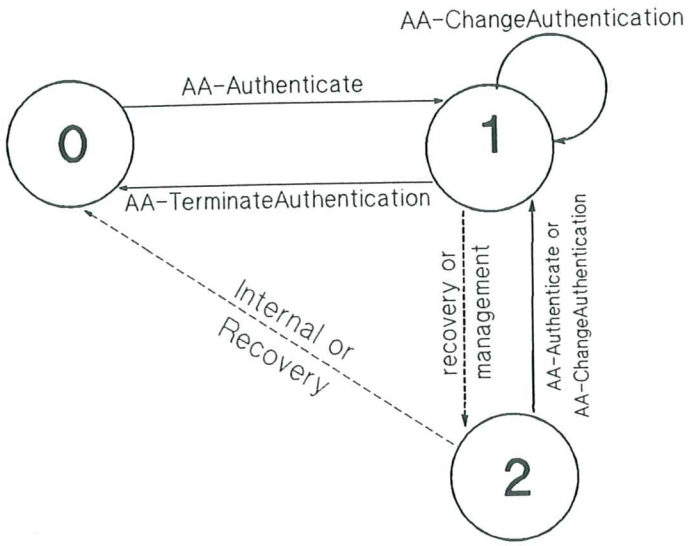

Figure 11 - Authentication State Changes and Service Primitives

4.2 Service Primitives

4.2.1 AA-Authenticate

The primitive to change from state 0 (not authenticated) to state 1 (authenticated) is the *AA-Authenticate* primitive. This primitive takes as its input parameter the authentication information of the entity, and returns the certified identity to be used in this session, or an error.

SERVICE	TYPE
AA-Authenticate	Confirmed
AA-ChangeAuthentication	Confirmed
AA-TerminateAuthentication	Confirmed
AA-CheckID	Confirmed
AA-Recovery	Confirmed
AA-Managememt	Confirmed

Table 2 - Authentication Service Primitives

The nature of the authentication information, and the choice of the actual standard protocol used to provide this information (2-way handshake, challenge-response, etc) will be determined by the security policy.

Parameter Name	Request	Response
Entity Authentication Credentials	M	
Authentication Method Identifier	U	
Role	U	
PAC (certified identity)		C
Authentication Level		C
Result		M
Error reason		C
Reject reason		C

Table 3 - AA-Authenticate Parameters

4.2.2.1 Entity Authentication Credentials

The information which the entity needs to supply to an Authentication Service. The form and nature of this information will be dependent on the authentication method chosen. This information will usually be confidential and will need protection (for instance, by a confidentiality service).

4.2.1.2 Authentication Method Identifier

Indicates which authentication procedure the entity wishes to invoke, where there may be a number of different procedures available to the entity.

4.2.1.3 Role

This parameter will give the role for which the entity wishes to be authenticated in the system. It is provided for two reasons:

1. to determine the level of authentication required;
2. to select an operational set of Privilege Attributes from the total set available following a successful authentication.

4.2.1.4 Privilege Attribute Certificate

This is the returned value of the service request if the request is successful (indicated by the result parameter). The returned PAC will contain identity attributes appropriate for the entity, role, and authentication level (indicating the type of authentication which has been carried out). It may also contain attributes associated with the path which the entity is using to access the system.

4.2.1.5 Authentication Level

This returned parameter shows the type of authentication the entity has carried out.

4.2.2 AA-ChangeAuthentication

This primitive uses the existing Certified Identity PAC, the required change, and any additional authentication information as input parameters and returns a new Certified Identity PAC or an error. This primitive may be used where the security policy requires re-authentication after some time period. An Authentication Service will need to update the *time of authentication* attribute in the Certified Identity. If re-authentication is not carried out in a timely way by the entity the authentication state may change from 1 (authenticated) to 2 (suspended) automatically. This primitive may also be used when an entity wishes to change some of the information in his Certified Identity PAC, for instance the authentication level or Privilege Attributes based on a previous role selection. This change may require additional authentication information (because the new role may have more privileges).

Parameter Name	Request	Response
PAC (certified identity)	M	
Authentication Method Identifier	U	
Role	U	
User authentication credentials	U	
PAC (certified Identity)		C
Authentication level		C
Result		M
Error reason		C
Reject reason		C

Table 4 - AA-Change-Authentication Parameters

4.2.2.1 PAC (Certified Identity)

This is the Certified Identity PAC returned from a previously successful AA-Authenticate request.

4.2.2.2 Authentication Method Identifier

Indicates which authentication procedure the entity wishes to invoke, where there may be a number of different procedures available to the entity.

4.2.2.3 Role

This is the new role which the entity wishes to assume within the system.

4.2.2.5 Entity Authentication Credentials

This is any additional authentication credentials which the entity must provide for the change of role and/or the change of authentication level.

4.2.2.5 PAC (Certified Identity)

If successful a new Certified Identity is returned to reflect the new role and authentication level. It may contain additional attributes as an efficiency measure.

4.2.2.6 Authentication Level

This returned parameter shows the type of authentication the entity has carried out.

4.2.3 AA-TerminateAuthentication

To tell the authentication service that the Certified Identity is no longer to be considered current the *AA-TerminateAuthentication* primitive is used. This primitive changes the entity's authentication state from authenticated (state 1) to not authenticated (state 0). The required input parameter is the entity's Certified Identity and the return is a confirmation or error.

Parameter Name	Request	Response
PAC (certified identity)	M	
Result		M
Error reason		C
Reject reason		C

Table 5 - AA-Terminate-Authentication Parameters

4.2.3.1 PAC (certified identity)

A PAC containing the Identity of the entity for which authentication is to be terminated.

4.2.4 AA-CheckID

Objects may need to check that a Certified Identity passed to them by another object is still valid, or that the entity is still acceptable to the system. The *AA-checkID* primitive may be used by any object to determine the current authentication status of an entity.

Parameter Name	Request	Response
PAC (certified identity)	M	
Authentication state		C
Additional information		C
Result		M
Error reason		C
Reject reason		C

Table 6 - AA-CheckID Parameters

The primitive will take the Certified Identity PAC as the input parameter and return a coded value of the current status (0-2). Where required by a security policy the value of the return may reflect additional state information maintained by the Authentication Service implementation.

4.2.4.1 PAC (certified identity)

The PAC (certified identity) of an entity that the application wishes to have checked.

4.2.4.2 Authentication State

The current state of the entity's authentication (see Operation Model above).

4.2.4.3 Additional Information

Additional information from an Authentication Service about the entity and the Authentication state.

4.2.5 AA-Recovery

If a security violation is detected in the system, or is suspected, then it may be necessary for the entity to be suspended or even have its session terminated. Two recovery operations are supported by an Authentication Service that will allow other security services to change the state of an entity as required by the security policy. The *suspend entity* operation will force a change from state 1 (authenticated and active) to state 2 (suspended). This primitive takes the identity of the entity as an input and returns a confirmation or error. Since the source of this recovery primitive is another security service it is not necessary to pass the certified identity of the entity to be suspended. Indeed the calling security entity may have no immediate mechanism for obtaining the current certified identity of the entity.

Parameter Name	Request	Response
Privilege attribute certificate	M	
Recovery operation	M	
Recovery information	U	
Result		M
Error reason		C
Reject reason		C

Table 7 - AA-Recovery Parameters

4.2.5.1 Privilege Attribute Certificate

The PAC under which the recovery primitive is to be authorised.

4.2.5.2 Recovery Operation

This parameter will indicate the appropriate recovery action, values will be assigned in a particular protocol. For an Authentication Service appropriate recovery operations will include: *suspend entity* (move to state 2) and *revoke entity authentication* (move to state 0). The *suspend service* and *resume service* operations to control the Authentication Service are also defined.

4.2.5.3 Recovery Information

Additional Information about the recovery action required, this may be used in a subsequent audit message from the invoked application.

4.2.6 AA-Management

An Authentication Service uses an authentication database to make its decisions. A number of service management operations are defined to allow external manipulation of this database. A

single service primitive is proposed for all security service management operations, it will be aligned with the security management services of the OSI management work.

Parameter Name	Request	Response
Privilege attribute certificate	M	
Management operation	M	
Management operation data	C	
Management information	U	
Result		M
Error reason		C
Reject reason		C

Table 8 - AA-Management Parameters

4.2.6.1 Privilege Attribute Certificate

The PAC which is to be used for access control associated with this service request.

4.2.6.2 Management Operation

A number of management operations have been identified, this list may not be exhaustive, combinations of basic operations may also be required. Consequently the actual operation is made a parameter as this can be extended in appropriate profiles. The current list of operations are:

- Add a new entry.
  This operation will allow new entities to be registered in the distributed system.
- Activate an Entry.
  This operation allows an entry in the database to be activated independently of its being registered, or after it has been suspended.
- Suspend an entry.
  This operation will allow entities to be temporarily stopped from using the system.
- Delete an entry.
  This operation allows entities to be permanently removed from the system.
- Update an entry.
  This operation will allow selective fields of a entity entry to be manipulated.
- Query an entry.
  This operation will allow an entry to be read out of the database, without change.

These operations will only be available to objects with the appropriate Privilege Attributes.

4.2.6.3 Management Operation Data

Any further parameters required by a particular operation.

5. SECURITY ATTRIBUTE SERVICE

Each service primitive has parameters for indicating the *result* of the service request (success, error, rejection, etc) and the reason for any *error* or *reject*. These parameters will be defined in detail in any protocols supporting the service primitive; they will not be described in any detail in this clause.

5.1 Operation Model

The primary role of a Security Attribute Service is that of mapping attributes. This includes providing more specific security attribute sets to be used with specific operations from initiators to targets, also providing and translating attributes for labelled objects.

SERVICE	TYPE
AT-GetPrivilegeAttributes	Confirmed
AT-TranslatePAC	Confirmed
AT-GetControlAttributes	Confirmed
AT-Recovery	Confirmed
AT-Management	Confirmed

Table 9 - Attribute Management Service Primitives

5.2 Service primitives

5.2.1 AT-GetPrivilegeAttributes

The *AT-GetPrivilegeAttributes* primitive receives the Certified Identity PAC and returns the appropriate set of Privilege Attributes in a second PAC.

Parameter Name	Request	Response
PAC (certified identity)	M	
Attribute Selection Criteria	M	
Privilege attribute certificate		C
Result		M
Error reason		C
Reject reason		C

Table 10 - AT-GetPrivilegeAttributes Parameters

5.2.1.1 PAC (certified identity)

The PAC returned from the AA-Authenticate service which will identify the entity requesting privilege attributes.

5.2.1.2 Attribute Selection Criteria

This parameter is used to select a subset of the total attributes available to the entity which should be included in the returned PAC.

5.2.1.3 Privilege Attribute Certificate

This is the returned item for a successful request. It contains the Privilege Attributes of the entity which fulfill the role and authentication level indicated by the Certified Identity PAC.

5.2.2 AT-TranslatePAC

Once the entity is established and has started some productive application there may arise a need to pass security attributes to other objects in the system. These attributes will be for a given operation to be performed by a given target for a given initiator, on behalf of the entity requesting the PAC. The attributes are bundled up into a certificate with some control information (see 3.2). An Attribute Service will build these sealed certificates under its own authority. Given the Privilege Attribute set of the entity (identified under the above primitive) and any necessary Control Attributes of the target object, an Attribute Service will provide the necessary Privilege Attribute Certificate. This is the *AT-TranslatePAC* primitive in which general attributes are mapped onto those required for a specific operation.

Parameter Name	Request	Response
Entity Privilege Attribute Certificate	M	
Initiator Control Attributes	U	
Target Control Attributes	U	
Attribute mapping criteria	M	
Privilege Attribute Certificate		C
Result		M
Error reason		C
Reject reason		C

Table 11 - AT-TranslatePAC Parameters

5.2.2.1 Entity PAC

A PAC returned in an AA-Authenticate or an AT-getAttributes service primitive or earlier AT-TranslatePAC call.

5.2.2.2 Initiator Control Attributes

These are the Control Attributes of the initiator which are to go in the initiatorQualifierAttributes field of the returned PAC (see 3.2.2.1).

5.2.2.3 Target Control Attributes

These are the Control Attributes for the target which are to go in the targetQualifierAttributes field of the returned PAC (see 3.2.2.1).

5.2.2.4 Attribute Mapping Criteria

This parameter is used to select a subset of the total attributes available to the requesting entity which should be included in the returned PAC.

5.2.2.5 Result Privilege Attribute Certificate

The returned PAC if the service primitive is successful.

5.2.3 AT-GetControlAttributes

An Attribute Service will also manage the Control Attribute data base for objects in the system. When one of these objects is created the *AT-GetControlAttributes* primitive will supply the appropriate Control Attribute set for the named object. For a newly created object the attributes returned will either have their values set to *uninitialised*, or to an initial default value determined by the local policy. If the engineering of the system uses the attribute database to contain the active values of Control Attributes for objects in the system then this primitive can be used to recover the actual values.

Parameter Name	Request	Response
Privilege attribute certificate	M	
Object identity	M	
Control attributes package		C
Result		M
Error reason		C
Reject reason		C

Table 12 - AT-GetControlAttributes Parameters

5.2.3.1 Privilege Attribute Certificate

The PAC under which access to the the Control Attributes are to be authorised.

5.2.3.2 Object Identity

The identity of the object for which the Control Attributes are required.

5.2.3.3 Control Attributes Package

The returned CAP if the service primitive is successful. The values ofthe Control Attributes will depend on local policy and on the use being made of the attribute database.

5.2.4 AT-Recovery

See 4.2.5 for details of the recovery primitive, only the values of the recovery action and information parameters will differ for this service. For an Attribute Service recovery actions might include: the revocation of an identity requiring that no PACs be allocated for that identity, revocation of a previously issued PAC, etc.

5.2.5 AT-Managementmanagement

The attribute service uses the attribute database to make its decisions. The operation of the *AT-Management* primitive is the same as that for AA-Management; only the values of the operation and information parameters are explicitly different. See 4.2.6 for the details of the primitive. A number of service management operations are defined to allow external manipulation of this database. The operations are:

- Add a new entry.
  This operation will allow new objects to be added to the system.
- Activate an Entry.
  This operation enables an entry to be activated independently of its being added to the database, or where it was previously suspended.
- Suspend an entry.
  This operation will allow objects to be temporarily suspended from the system.
- Delete an entry.
  This operation allows objects to be permanently removed from the system.
- Update an entry.
  This operation will allow selective fields of an attribute mapping entry to be manipulated.
- Query an entry.
  This operation will allow an entry to be read out of the database, without change.

These operations will only be accessible to objects with the appropriate Privilege Attributes.

6. SECURE ASSOCIATION SERVICE

Each service primitive has parameters for indicating the *result* of the service request (success, error, rejection, etc) and the reason for any *error* or *reject*. These parameters will be defined in detail in any protocols supporting the service primitive; they will not be described in any detail in this clause.

6.1 Operations Model

The purpose of this service is to allow applications to initiate, control and release secure associations with other objects in a distributed system. It includes the authorisation of these associations. The relationship between a Secure Association Service and the BIND operation used in some application layer protocols is for further study (but see also clause 10).

A Secure Association Service may in the future also be responsible for policing the flow of data across the associations it creates, this is a matter for further study.

A Secure Association Service incorporates the mechanisms for access control between objects. To achieve this it calls an Authorisation Service, with the Privilege Attribute Certificate of the initiator and other information. A Secure Association Service requires the called, and sometimes the calling, object titles (identities). The access control mechanisms invoked when initiating a secure association can equally well be applied when releasing the association. Aborting a secure association should be reported as a major security event as the normal procedure should be a proper release.

To establish the required security on an association the service uses the Target Security Parameters (see ISO 7498/2). The service makes no assumption about how the required security is provided by the OSI services supporting the application-entity association.

The target and initiator components of a Secure Association Service may carry out access control and invoke an Authorisation Service to accept or reject association requests.

Some part of a Secure Association Service will need to be closely integrated with the security infrastructure on an end-system to ensure enforcement of access control and to support security mechanisms in the application layer, such as peer entity authentication. This relationship is for further study.

SERVICE	TYPE
SA-Initiate	Confirmed
SA-Release	Confirmed
SA-Abort	Unconfirmed
SA-Recovery	Confirmed
SA-Management	Confirmed

Table 13 - Secure Association Service Primitives

6.2 Service Primitives

6.2.1 SA-Initiate

The *SA-Initiate* primitive is used to open an association between two objects in a secure manner.

6.2.1.1 Calling Application Identity

The identity of the calling object.

6.2.1.2 Called Application Identity

The identity of the called object.

Parameter Name	Request	Response
Calling Application Identity	U	
Called Application Identity	M	
Privilege Attribute Certificates	M	
Target Security Parameters	U	
Result		M
Error reason		U
Reject Reason		U

Table 14 - SA-Initiate Parameters

6.2.1.3 Privilege Attribute Certificates

These will be used by a Secure Association Service in both the calling and called end-systems to apply access control in the association establishment phase.

6.2.1.4 Target Security Parameters (TSP)

The security which the calling object requires on the association. Appropriate values are: confidentiality, integrity and peer-entity authentication (see ISO 7498-2 sub-clause 7.8). Where a security policy requires, certain security parameters may be applied without being requested.

6.2.2 SA-Release

At any time the initiator and target can request the Secure Association Service to release the association in a controlled manner.

Parameter Name	Request	Response
Privilege Attribute Certificate	M	
Reason	U	
Result		M
Error Reason		C
Reject Reason		C

Table 15 - SA-Release Parameters

6.2.2.1 Privilege Attribute Certificate

This may be used by a Secure Association Service in both the initiator and target end-systems to apply access control to the association release phase.

6.2.2.2 Reason

The reason for requesting the release of the association. This information is conveyed to the other end-point.

6.2.3 SA-Abort

This primitive allows a secure association to be aborted.

Parameter Name	Request	Response
Reason	U	
Result		M
Error Reason		C
Reject Reason		C

Table 16 - SA-AbortParameters

6.2.3.1 Reason

The reason for aborting the association which may be conveyed to the other end-point.

6.2.4 SA-Recovery

This service element is used by a security recovery or management entity to abort those associations which are a threat to the security of the system. The secure association service provider may create an audit log entry for this event.

See 4.2.5 for details of the recovery primitive, the *abort* function is defined for the SA-Recovery primitive.

6.2.5 SA-Management

This subject is closely related to OSI Management and is for further study.

7. AUTHORISATION SERVICE

Each service primitive has parameters for indicating the *result* of the service request (success, error, rejection, etc) and the reason for any *error* or *reject*. These parameters will be defined in detail in any protocols supporting the service primitive; they will not be described in any detail in this clause.

7.1 Operation Model

An Authorisation Service provides access control decision making for other security services and objects. Enforcement of these decisions is performed by the security infrastructure and outside the scope of this ECMA Standard.

An Authorisation Service will require a ruleset to derive its decisions, it may also utilise local context information as required the local security policy.

SERVICE	TYPE
AU-Decision	Confirmed
AU-Recovery	Confirmed
AU-Management	Confirmed

Table 17 - Authorisation Service Primitives

7.2 Service Primitives

7.2.1 AU-Decision

The authorisation decision is based on given Privilege and Control Attributes, and context derived information. The result of an authorisation decision is either *yes*, *no* or *error*.

Parameter Name	Request	Response
Privilege attributes	M	
Control attributes package	M	
Operation description	M	
Access Context	U	
Decision		C
Result		M
Error Reason		C
Reject Reason		C

Table 18 - AU-Decision Parameters

7.2.1.1 Privilege Attributes

These may be passed either as a simple list of attributes or in the form of a PAC. In either case an Authorisation Service may refer to an Attribute Service for mapping as required by the security policy.

7.2.1.2 Control Attributes Package

The Control Attributes of the object to be accessed. An Authorisation Service may refer them to an Attribute Service for mapping as required by the security policy.

7.2.1.3 Operation Description

A description of the operation to be carried out.

7.2.1.4 Access Context

Any additional contextual information which may be necessary for the decision.

7.2.1.5 Decision

The result of the authorisation decision.

7.2.2 AU-Recovery

The Recovery service interface is described in 4.2.5. The recovery operation defined for an Authorisation Service are: *refuse to accept PACs with certain recoveryIdentifiers.*

7.2.3 AU-Management

The management primitive is described in detail in 4.2.6. Specific operations for an Authorisation service are described here. An Authorisation Service uses the authorisation ruleset to make its decisions. A number of service management operations are defined to allow external manipulation of the ruleset. The operations are:

- Add a new entry.
  This operation will allow new access control rules to be added to the system.
- Activate an entry.
  This operation will allow a rule entry to be activated independently of its being added to the database, or if an entry has previously been suspended.
- Suspend an entry.
  This operation will allow access to be temporarily suspended for a selected User, or object.
- Delete an entry.
  This operation allows an access to be permanently removed from the system.
- Update an entry.
  This operation will allow selective fields of an access entry to be manipulated.
- Query an entry.
  This operation will allow an entry to be read out of the ruleset, without change.

These operations will only be accessible to objects with the appropriate Privilege Attributes.

8. INTERDOMAIN SERVICE

Each service primitive has parameters for indicating the *result* of the service request (success, error, rejection, etc) and the reasonfor any *error* or *reject*. These parameters will be definedin detail in any protocols supporting the service primitive; theywill not be described in any detail in this clause.

8.1 Operation Model

Establishing secure associations between objects in differentsecurity domains requires the use of Interdomain Services in one or more of the domains to map security attributes. The mapped attributes may have to be resealed so that they will be accepted in the target domain. There is no real-time communication between the Interdomain Services involved.

SERVICE	TYPE
ID-TranslateAttributes	Confirmed
ID-Recovery	Confirmed
ID-Management	Confirmed

Table 19 - Interdomain Service Primitives

This service translates PACs and labelled objects. The appropriate mapping will be a subject of domain policy. PACs and labelled objects passing between domains will contain attributes with an agreed interdomain attribute syntax, such as that described in 3.7.

8.2 Service Primitives

8.2.1 ID-TranslateAttributes

This service request supports the translation of extra-domain attributes to their intra-domain equivalents. The new attributes will be sealed by an authority recognised by the authentication service of the local domain. The original Privilege Attribute Certificate may be embedded in the new PAC.

Parameter Name	Request	Response
Attributes to be translated	M	
Attribute mapping criteria	M	
Remote domain name	M	
Translated attributes		C
Result		M
Error Reason		C
Reject Reason		C

Table 20 - ID-MapAttributes Parameters

8.2.1.1 Attributes to be translated

This is a PAC or labelled object, the contents of which need to be translated.

8.2.1.2 Attribute Mapping Criteria

This parameter can be used to apply constraints and helpful information for the translation process.

8.2.1.3 Remote Domain Name

The name of the remote domain and its authority.

8.2.1.4 Translated Attributes

The new PAC or labelled object with the translated attributes.

8.2.2 ID-Recovery

The Recovery service interface is described in 4.2.5. The recovery operation defined for the Interdomain Service are: *refuse to accept PACs with certain PAC Identifiers.*

8.2.3 ID-Management

The Interdomain Service maintains a database of information necessary to perform its mapping and sealing operations. The details of the management primitive are given in 4.2.6 The operations defined for the Interdomain Service are:

- Add a new entry.
  This operation will allow new mappings to be added to the database.
- Activate an entry.
  This operation allows an entry to be activated independently of its addition to the database, or if it has previously been suspended.
- Suspend an entry.
  This operation will allow mappings to be temporarily suspended.
- Delete an entry.
  This operation allows a mapping to be permanently removed from the database.

- Update an entry.
  This operation will allow selective fields of a mapping entry to be manipulated.
- Query an entry.
  This operation will allow an entry to be read out of the database, without change.

These operations will only be accessible to objects with the appropriate Privilege Attributes.

9. SECURITY AUDIT INFORMATION COLLECTION SERVICE

Each service primitive has parameters for indicating the *result* of the service request (success, error, rejection, etc) and the reason for any *error* or *reject*. These parameters will be defined in detail in any protocols supporting the service primitive; they will not be described in any detail in this clause.

9.1 Operations Model

The Security Audit Service records significant security-related events. The specific events for which an audit record is required are specified to each individual security service by the security manager in line with the security policy defined for that domain.

The Security Audit Information Collection Service supports the general interface used by other security services, and applications, to submit audit information to a repository. The analysis and subsequent actions of information in the repository will be determined by the domain security policy. Only one primitive is required in this service. The management interface of the security services and objects will be used to adjust the selection of information to be audited.

SERVICE	TYPE
AS-Audit-Information-Collection	Confirmed

Table 21 - Audit Service Primitives

9.2 Service Primitives

9.2.1 AS-Audit-Information-Collection

This service is used by all other security services, and objects in the system, to record audit information for subsequent analysis. The only responsibility of this service to accept the audit information securely and maintain it until processed. Analysis and subsequent storage of audit information is dependent on the local security policy.

Parameter Name	Request	Response
Privilege Attribute Certificate	M	
Audit Information	M	
Result		M
Error Reason		C
Reject Reason		C

Table 22 - AS-Audit Parameters

9.2.1.1 Privilege Attribute Certificate

The PAC of the application sending the Audit information.

9.2.1.2 Audit Information

The information to be recorded by the service. Specification of the information is outside the scope of this ECMA Standard.

10. USING THE SECURITY MODEL

This clause shows applications standards writers how to define the security requirements of their standard in a way which allows the applications concerned to interwork with other standard applications (including specialist security applications) in a distributed system.

In any standard being developed it is important for developers to identify the design freedoms that need to be restricted to allow interworking (the subject of the specific standard) and design freedoms that should be left to the users of the standard. In the case of security, because different Security Administrations have widely different policy requirements, the design freedoms that need to be left are considerable.

In this ECMA Standard the only specific constraints imposed on any user of the standard are the format in which security information is to be transferred and the occasions on which it is appropriate to transfer it. Security Policy will always be left as a design freedom, so the actual security information to be carried has not been defined (though some standard types are offered). The security information defined in this ECMA Standard can be used to implement policies based on Access Control Lists, Security Labels, Capabilities and many other control mechanisms along this spectrum.

The next part of this clause describes common aspects of access control, the use of Authentication Services and Attribute Services and their relationship to Secure Association Services. The remaining parts then describe how application standards are affected by the other security services of Audit, Recovery, Interdomain, and Authorisation.

This clause is written using "application and process" terminology, not the terminology of the object oriented model. This is because the clause is aimed at applications standards writers, especially those working on OSI applications. It also shows that the object model of security is easily translated into this terminology with no loss of information.

10.1 Common Aspects of Access Control

This sub-clause is structured as follows: first an overview is given of the aspects of application standards definition that are affected; this is followed by a detailed description of each aspect in turn, outlining the syntactic elements that are affected; finally all of the changes are brought together in some example ASN.1.

10.1.1 Overview

Application standards are being defined for applications which are to be usable and manageable in the context of large distributed systems supporting multiple applications of different types. Such standards must therefore be developed to take account of this context.

In particular a number of aspects of access control should be dealt with in the same manner by all application standards writers; they are to do with security subjects accessing the system, the globally understood security privileges that belong to them and security control attributes of protected objects as they are communicated between end systems.The related aspects of access control are:

- Subject authentication (this may be on the basis of a warranty by some trusted authority external to the application).
- Acceptance of subject access privileges provided by a trusted authority external to the application. There are two kinds of these: those required of the subject by the access control authorisation logic in the accessed application itself, and those that the application may have to acquire to pass on to further applications whose support is required in the performance of the action requested by the initiator.

- Submission of evidence of an application's own access privileges for use in accessing other applications and the objects they control.
- Communication of security control attributes when creating an object or revising or interrogating an object's security control attributes, or when communicating diagnostic or security management information.
- Construction and attachment of security data to objects for external transmission.

Each of these aspects is dealt with below, the first three together.

10.1.2 Authentication and Access Privileges

As users of computer systems become more sophisticated, and as more activities in business begin to involve computers, each individual user will typically require to use an increasing number of applications, possibly in parallel. Also, the emergence of a variety of access control models based on access privileges other than simple user identity, coupled with the need in large distributed systems to manage these privileges independently of particular applications has led to a requirement to enrich a user's identity with additional data which defines these privileges.

The ECMA security model, described in clause 2, enables a human user to authenticate himself only once to the system, not individually to each application. As a result of the authentication the user will have Privilege Attributes that can be used with all applications in the system for access control and identifying the user for other purposes (such as charging). This ECMA Standard defines the syntax and methods for exchanging Privilege Attributes. Similarly, applications can be authenticated and may possess audit identities and Privilege Attributes which can be used within the system for application interaction. In following this model, application standards designers should use a common mechanism for offering and accepting Privilege Attributes.

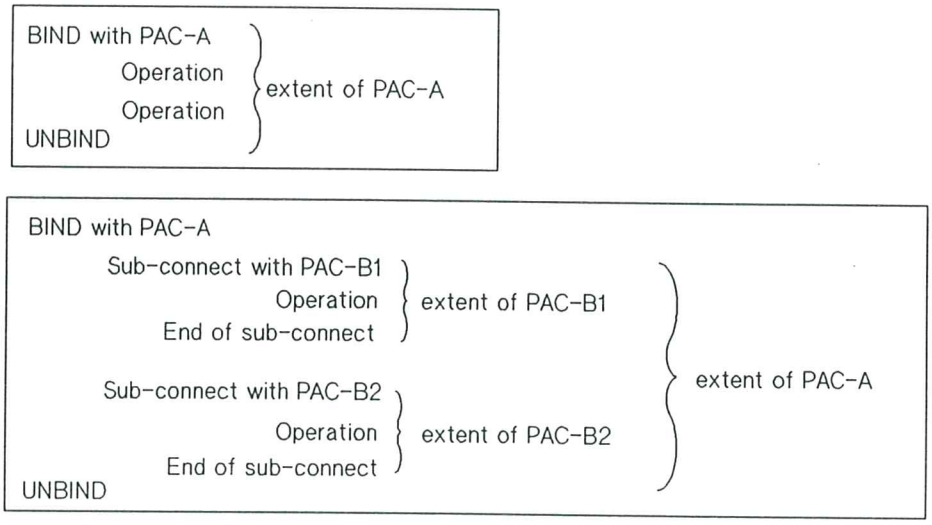

Figure 12 - PAC Passing and Usage Alternatives

External support for providing a target application with the certified identities and other access privileges of its users can be given in two ways:

1. The application can depend on a co-located trusted infrastructure (for example its host operating system) to provide it with the accessing subject's identity and other privilege attributes, obtained by the infrastructure in a PAC, in a manner similar to that described for the application itself below. The infrastructure will have checked the validity of the PAC, will have unpacked it, and may also have performed some mapping of the Privilege Attributesprivilege attribute found in the PAC, into locally understood, possibly application-specific values.

2. The application can obtain the accessing subject's identity and other privilege attributes directly via a PAC.

In the first case, application access protocols are affected only in that no authentication is visible; application standards designers should therefore permit this as an option. Standardisation of the interface between the application and its host infrastructure is primarily an application portability issue, not an interworking standards issue.

In the second case, the PAC may be passed on any of three occasions, depending on the control relationship between the accessing subject and the association resulting from a BIND operation. They are as follows:

1. When each accessing subject initiates an association with the target application via a BIND which is dedicated to that subject's use, a PAC representing some or all of the subject's access rights is passed as a BIND argument (PAC-A in figure 12).

2. When a single association is multiplexed between different accessing subjects (for example in some transaction processing environments), then if there is a concept, probably application-type specific, of a within-bind "sub-connection" the subject's PAC will be passed as an argument to the "sub-connect" operation (PAC-B1, PAC-B2 in figure 12); otherwise it can be passed either with the first operation for each user (treating it effectively as an implicit sub-connect) or with every operation, as in the connectionless relationship described next.

3. When a connectionless mode of access to the application is supported, a PAC is presented with each operational command (PAC-C1 PAC-C2, PAC-C3 in Figure 12).

10.1.2.1 Specification in an Application Standard

The specification of a BIND operation should be structured so that application-specific arguments are easily separable from those used for common aspects of access control. The security-related common arguments identified so far for a BIND operation are:

- a PAC to establish the access privileges for operations performed in the context of the BIND. This assumes external authentication by a trusted authority has already taken place;
- nothing, assuming the infrastructure support described above;
- authentication information presented to enable the application to authenticate the accessing subject (this is not consistent with this ECMA Standard but may be required for systems in which no external support for authentication is provided).

The specification of a "sub-connect" operation should include the optional presentation of a PAC to establish access privileges for operations performed in the context of the sub-connection.

The specification of other kinds of operation (including "sub-connect" operations) should include the optional presentation of a PAC containing (additional) attributes required for the operation.

Use of all of three methods in combination is possible.

10.1.3 Proxy and Compound Subjects

Some operations requested of an application are not performed directly or completely by that application, but are wholly or partially "sub-contracted" out to another application; for example a text editing application may use a file server to hold files of text, and when a user asks to start an edit, the editor must ask the file server for the user's file. This is a case of *Proxy*; the editor is acting with respect to the file server as the user's proxy, and it must convince the file server of its right to access the file. Depending on the trust relationship between the two applications (which is determined by the security policy in force); it may do this in a number of ways, for instance:

1. The editor may be trusted to ask only for files that the user may legitimately edit. In this case the editor acts as the only accessing subject; the file server sees only the editor's access rights and expects a PAC from it, which belongs to it, and which is presented in one of the ways described in the previous clause.

2. The editor may have no access rights of its own to the files in the file server, and must prove to the file server that it has permission from the user to obtain the file. It does this by passing a *Proxy PAC* to the file server as a parameter to the access request for the file. The Proxy PAC belongs to the user, and was obtained from the user parametrically either with the user's operation request or via a previous operation. The proxy PAC may have been constructed to tightly constrain the editor's access (for example it may permit read access only to a particular named file)

3. As above, the editor has no rights of its own but a *Referenced Data Transfer* (see ECMA-131 and ISO 10031/2) facility is supported by the file server, in which the user warns the file server in advance that the editor will be making the access request, and pre-authorises the access. This is for further study.

4. The security policy in force takes into account both the access rights of the user and the access rights of the editor (for example the user may be permitted access to the file only if it uses the editor to access it, and the editor has access only if it is acting for the user). In this case a combination of 1. and 2. or 1. and 3. above occurs. The file server is being accessed by a *Compound Subject.*

10.1.3.1 Privileges Required to Perform a BIND

When an association is being formed between two applications on behalf of a user, the access privileges required in order to make the BIND succeed may be different from those which will be established for the user for operations within the association. In particular, the user may not possess sufficient authority to form the association, and the relevant Secure Association Service may require an additional PAC (e.g. from the initiating application) to be passed to it as a BIND argument.

10.1.3.2 Specification in an Application Standard

The application standard writer cannot assume any particular trust relationship between two applications, since this will vary according to security policy. Therefore:

- for operations that may be forwarded to a second application, the first application should be capable of accepting a Proxy PAC to be passed by the originator of the

request, for use by it as the required operational PAC in its operation call on the second application;

- when an initiator BINDs to a target it must be able to pass a PAC representing the access privileges required to permit the establishment of the association as well as one for its controlling user.

10.1.4 Secure Association Service

This service provides for secure communications between applications, but needs to be informed of the nature of the (ISO) security services wanted. This information must be passed when the association between the applications is formed. The other responsibilities of the Secure Association Service do not impact on the specification of application standards, except indirectly as described in the previous sub-clause.

10.1.5 Example ASN.1

The following ASN.1 is offered as an example of changes to interface definitions required to support the common aspects of access control.

```
Bind-Argument ::= SEQUENCE{
    application-specific-arguments    [0] Application,
    common-security-arguments         [1] Security}

    -- Application ::= whatever the application requires

Security ::= SEQUENCE {
        CHOICE {
    credentials-for-appl              [0] Creds,
    certified-id-and-privileges       [1] Bind-PACs} OPTIONAL,
        bind-security ServicesRequired}

Bind-PACs ::= SEQUENCE SIZE (1 ... 2) {
    required-bind-PAC                 [0] PrivilegeAttributeCertificate OPTIONAL,
    context-PAC                       [1] PrivilegeAttributeCertificate OPTIONAL}

Sub-Connect-Argument ::= SEQUENCE {
    sub-connect-specific-argument     [0] Sub,
    common-security-argument          [1] Context-PAC }

    -- Sub ::= whatever the sub-connect requires

Operation-X-Argument ::= SEQUENCE{
    operation-specific-argument       [0] Op,
    common-security-argument          [1] Op-Sec}

    -- Op  ::= whatever the operation requires

Op-Sec ::= SEQUENCE{
    required-operation-PAC            [0] Operation-PAC OPTIONAL,
    proxy-PAC                         [1] Proxy-PAC OPTIONAL}
```

NOTE 7

credentials-for-appl *is used by the target to authenticate the initiator. This use is not consistent with this ECMA Standard, but is included for backwards compatibility.*

NOTE 8

bind-security *is used to tell the Secure Association Management Service what kind of security protection is needed on the association that will result from the BIND.*

NOTE 9

required-bind-PAC *is the PAC used to present the privilege attributes required to perform the BIND.*

NOTE 10

context-PAC *is the PAC used to present the privilege attributes to be associated with operations performed in the context of the BIND or "sub-connect".*

NOTE 11

required-operation-PAC *is the PAC passed with an operation when the access privileges currently established are insufficient to permit the requested operation. This may happen when either the operation requires special additional privileges, or when a BIND is being multiplexed between multiple users.*

NOTE 12

proxy-PAC *is used when the target needs to make further accesses to another application on behalf of the calling subject, and itself would have insufficient access rights unless supplemented by those in the* proxy-PAC. *The first target will itself then use this PAC as an operation-PAC when it acts as an initiator in its call on the next target.*

10.1.6 Object Control Attributes

When an application object is created the security Control Attributes that are used to protect it from unauthorised access need to be defined and established. The exact way in which this is done will be particular to the security policy in force, but under some policies some of these attributes will be specifiable as arguments to the "Create" operation. An Attribute Service may be used to support this function as described in clause 5. When the object is accessed its control attributes may need to be read or written.

The syntax of a Security Attribute is defined in clause 3 to be the same as other kinds of object attribute and an application need make no distinction between the two kinds, except in the following respects:

- the standards developer should clearly separate out any rules that may be specified about what Security Attribute values are acceptable in what contexts, so that the security policies supported are in this respect are clearly identifiable;
- these rules should be specified as optional to allow for variations so that the application can be used in different security policy contexts;
- the definition of the types of Control Attribute that are to be supported should be similarly flexible and easily separable;
- the standards developer should consider the advantages of separating out operation arguments containing Security Attributes from other arguments to simplify any functionality that an implementor might provide to protect them and control their use.

10.1.7 Labelled Objects

Clause 3.6 introduces the concept of a labelled object, an object and its security attributes - formatted according to some defined syntax - bound together in a single data structure under the seal of an identified security authority.

Labelled objects, because of their standardised syntax, are suitable for interchange between different open systems whereas application specific objects are not suitable and need special conversion operations. However, application specific security policies may require the use of

security attributes that have relevance only to the application environment. Where such objects need to be transfered, or stored, by other applications a commonly agreed syntax of security attributes is needed if other applications are to safeguard the security of these objects during transmission and storage. The standard syntax for a labelled object as given in 3.5.1 provides this common syntax.

Conversion from internal security attribute syntax, and vice versa, is the responsibility of the applications manipulating the protected objects. A Security Attribute Service and an Interdomain Service provide primitives for creation and translation under the control of an application.

An example may make the use of Label translation clear: assume a protected document, held on a secure Document Filing and Retrieval Application. The document object contains security attributes specific to the DFR application. If the document has to be transmitted by electronic mail in a secure manner, a subset of its attributes must be copied and translated to a security attribute syntax that can be understood by - but need not be specific to - the E-mail application. The latter can then treat the document according to the policy elements identified by the label content. Where the document is received, it may be put into a file server. Again the standard syntax of the security attributes in the label allow the file server to treat the document correctly.

10.2 Audit

Auditors require that a record of security significant events be produced, but although support of audit is a universal requirement in virtually all secure systems it is not an interworking issue, and applications implementors should be free to choose their own approach to audit. They are recommended to allow flexibility of management of selection of auditable events.

This document introduces a standard Audit Service. Applications implementors will in future be advised to permit their Audit Managers to direct audit messages to it. Use of the standard Audit Service may relieve application designers of a number of functional support requirements such as audit trail protection, archive and analysis, and provides a focal point for cross correlation of audit data from multiple sources in a distributed system.

10.3 Recovery

In most cases the security recovery function is an integral part of the application. Standards developers should consider security risks to their applications and specify appropriate recovery procedures within their standard.

10.4 Attribute and Interdomain Service

The access control policy of an application will commonly be partly, if not wholly, defined in terms of Privilege Attributes that have semantics and/or representation that are local to the application. Therefore when an application receives a PAC containing global attributes it may require a mapping to be made between these global attributes and locally understood values. This is one of the services of an Attribute Service. Whether this is viewed by the application as being a component of an Interdomain Service or whether the application sees an Attribute Service directly is a matter for further study; both options may apply, depending on policy and on the relationships that exist between attribute authorities. In either case the important principle is that the task of Privilege Attribute mapping is identified as a separable function which will operate in different ways according to the security context within which the application is being run.

When an application is releasing an object for transmission, as described in 10.1.7, it will use an Attribute Service, possibly via an Interdomain Service as described above, to map the object's local Control Attributes to the standard forms required within the transmission security label.

10.5 Authorisation

An application is responsible for authorising access to its own objects. It may optionally choose to control access to itself, though in many environments this control is provided by a Secure Association Service. Authorisation of access to an application's objects is performed by an Authorisation Service, which needs information on which to base its decisions; these are the Privilege and Control Attributes described in clause 3, coupled with information about the operation and the access context pertaining at the time of the access requests.

It is an important principle of this ECMA Standard that access authorisation should be viewed in this separate way, even though an Authorisation Service may be implemented by the application programmer as a part of the application. This approach is necessary in order that the same basic application functionality can be used within different access control policy contexts, by offering alternative Authorisation Services (or clearly identifiable different management configuration options of the same Authorisation Service) for each context. In this way the application standards developer can describe Authorisation Service policies that are particularly suitable, without permanently limiting the security contexts within which the application can be used.

APPENDICES

Appendix A
Abstract Syntax

This Appendix is part of the Standard

This Appendix contains the ASN.1 definitions in their complete form.

Security-Service-Notation { iso(1) identified-organization(3) icd-ecma(0012) standard(0) desd(138) notation(0) }

DEFINITIONS ::=

-- Module contains Object Identifiers and Macros used in subsequent modules

BEGIN

-- Exports Everything

IMPORTS; *-- Nothing*

ID ::= **OBJECT IDENTIFIER**

-- root for DESD object identifier value

id-desd ID ::= { iso(1) identified-organization(3) icd-ecma(0012) standard(0) desd(138) attributeIdentifiers(3) }

-- Object Identifiers for the Security Attribute types

desd-att-accounting-identity	ID ::= { id-desd 1 }
desd-att-audit-identity	ID ::= { id-desd 2 }
desd-att-authentication-level	ID ::= { id-desd 3 }
desd-att-capability-type-1	ID ::= { id-desd 4 }
desd-att-confidentiality-class	ID ::= { id-desd 5 }
desd-att-confidentiality-class-minimum	ID ::= { id-desd 6 }
desd-att-confidentiality-hierarchy	ID ::= { id-desd 7 }
desd-att-confidentiality-hierarchy-minimum	ID ::= { id-desd 8 }
desd-att-entity-identity	ID ::= { id-desd 9 }
desd-att-group	ID ::= { id-desd 10 }
desd-att-integrity-class	ID ::= { id-desd 11 }
desd-att-integrity-class-minimum	ID ::= { id-desd 12 }
desd-att-integrity-hierarchy	ID ::= { id-desd 13 }
desd-att-integrity-hierarchy-minimum	ID ::= { id-desd 14 }
desd-att-need-to-know	ID ::= { id-desd 15 }
desd-att-need-to-know-minimum	ID ::= { id-desd 16 }
desd-att-role	ID ::= { id-desd 17 }

END *-- of Notation Module*

```
Security-Service-Information { iso(1) identified-organization(3) icd-ecma(0012) standard(0)
desd(138) securityData(1) }

DEFINITIONS ::=
-- Module contains the definitions of the Security Data Elelments
BEGIN

EXPORTS
   PrivilegeAttributeCertificate, LabelledObject, ControlAttributePackage;

IMPORTS
   Security-audit-identity, Security-acounting-identity FROM
   Security-Service-Attributes { iso(1) identified-organization(3) icd-ecma(0012)
   standard(0) desd(138) securityAttributes(2) };

-- definition of the Privilege Attribute Certificate

-- this is the sealed unit which is sent down the wire
PrivilegeAttributeCertificate ::= SEQUENCE {
   SEQUENCE {
      -- identifies the version of the PAC syntax
      syntaxVersion INTEGER (version1 (1)),

      -- for a compound PAC:
         containedPACs          [0] SEQUENCE OF PrivilegeAttributeCertificate OPTIONAL,

      -- the actual privilege attributes:
      privilegeAttributes       [1] SEQUENCE OF SecurityAttribute,

      -- fields used to support PAC protection schemes
      initiatorQualifierAttributes [2] SEQUENCE OF SecurityAttribute OPTIONAL,
      targetQualifierAttributes [3] SEQUENCE OF SecurityAttribute OPTIONAL,
      validationKeyIdentifier   [4] BIT STRING OPTIONAL,

      -- fields for additional protection in use
      creationTime              [5] UTCTime OPTIONAL,
      validityTime              [6] TimePeriod OPTIONAL,
      pacIdentifier             [7] INTEGER OPTIONAL,
      recoveryIdentifier        [8] INTEGER OPTIONAL,

      -- Identity fields for use by intermediate handlers
      auditIdentifier           [9] Security-audit-identity OPTIONAL,
      chargingIdentifier        [10] Security-accounting-identity OPTIONAL,

      -- fields used for the certification:
      pacAuthority              [11] Authority },
   -- seal on the end to assist length calculations
   pacSeal Seal }

-- Control Attribute Package for transferring Control Attributes
ControlAttributePackage ::= SEQUENCE OF SecurityAttribute
```

```
SecurityAttribute ::= SEQUENCE {
   attributeType    OBJECT IDENTIFIER,
   attributeValues  SEQUENCE {
            authority [0] PolicyAuthority OPTIONAL,
      SEQUENCE OF ANY }} -- DEFINED BY attributeType

   PolicyAuthority ::= OBJECT IDENTIFIER

-- Certificate encapsulating an object with its Control Attributes
LabelledObject ::= SEQUENCE {
   objectType       OBJECT IDENTIFIER,
   object           ANY DEFINED BY objectType,
   labelAttributes  SEQUENCE OF SecurityAttribute,
   labelSeal Seal,
   labelAuthority Authority }

Authority ::= SEQUENCE { -- to permit a variety of naming schemes
   authorityType    OBJECT IDENTIFIER,
   authorityValue   ANY DEFINED BY authorityType }

Seal ::= SEQUENCE {
   algorithmIdentifier  OBJECT IDENTIFIER,
   keyIdentifier        OCTET STRING,
   seal                 OCTET STRING }

TimePeriod ::= SEQUENCE OF SEQUENCE {
   startTime[0] UTCTime OPTIONAL,
   endTime [1] UTCTime OPTIONAL }

END -- of Security Information Module

Security-Service-Attributes{ iso(1) identified-organization(3) icd-ecma(0012) standard(0)
desd(138) securityAttributes(2) }

DEFINITIONS ::=
-- Module contains the definitions of the Security Attributes
BEGIN

EXPORTS
   Security-accounting-identity, Security-audit-identity, Security-authentication-level,
   Security-capability-type-1, Security-confidentiality-class,
   Security-confidentiality-class-minimum, Security-confidentiality-hierarchy,
   Security-confidentiality-hierarchy-minimum, Security-entity-identity, Security-group,
   Security-integrity-class, Security-integrity-class-minimum, Security-integrity-hierarchy,
   Security-integrity-hierarchy-minimum, Security-need-to-know,
   Security-need-to-know-minimum, Security-role;
```

```
IMPORTS
  -- ATTRIBUTE macro from X.500
  ATTRIBUTE FROM InformationFramework {joint-iso-ccitt ds(5) modules(1)
  informationFramework(1)}

  -- Object Identifiers
  desd-att-accounting-identity, desd-att-audit-identity, desd-att-authentication-level,
  desd-att-capability-type-1, desd-att-confidentiality-class,
  desd-att-confidentiality-class-minimum, desd-att-confidentiality-hierarchy,
  desd-att-confidentiality-hierarchy-minimum, desd-att-entity-identity, desd-att-group,
  desd-att-integrity-class, desd-att-integrity-class-minimum, desd-att-integrity-hierarchy,
  desd-att-integrity-hierarchy-minimum, desd-att-need-to-know,
  desd-att-need-to-know-minimum, desd-att-role FROM
  Security-Service-Notation { iso(1) identified-organization(3) icd-ecma(0012) standard(0)
desd(138) notation(0) };

-- Definition of common data structure to avoid duplication of attribute definitions

  IntegerOrString ::= CHOICE {
    integerPart INTEGER,
    stringPart IA5String }

-- Definitions of common standard security attributes

  Security-accounting-identity ::= ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX IntegerOrString
    SINGLE VALUE

  Security-audit-identity ::= ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX IntegerOrString
    SINGLE VALUE

  Security-authentication-level ::= ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX INTEGER
    SINGLE VALUE

  Security-capability-type-1 ::= ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX CapabilityType1
    SINGLE VALUE

  CapabilityType1 ::= SEQUENCE {
    objectDefiner ObjectDefiner,
    accessType AccessDefiner}

  ObjectDefiner ::= IntegerOrString
  AccessDefiner ::= IntegerOrString

  Security-confidentiality-class ::= ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX PrintableString
    MULTI VALUE
```

Security-confidentiality-class-minimum :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX PrintableString
SINGLE **VALUE**

Security-confidentiality-hierarchy :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX **INTEGER**
SINGLE **VALUE**

Security-confidentiality-hierarchy-minimum :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX INTEGER
SINGLE **VALUE**

Security-entity-identity :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX IntegerOrString
SINGLE **VALUE**

Security-group :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX IntegerOrString
MULTI **VALUE**

Security-integrity-class :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX PrintableString
MULTI **VALUE**

Security-integrity-class-minimum :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX PrintableString
MULTI **VALUE**

Security-integrity-hierarchy :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX **INTEGER**
SINGLE **VALUE**

Security-integrity-hierarchy-minimum :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX **INTEGER**
SINGLE **VALUE**

Security-need-to-know :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX PrintableString
MULTI **VALUE**

Security-need-to-know-minimum :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX PrintableString
MULTI **VALUE**

Security-role :: = ATTRIBUTE
WITH ATTRIBUTE-SYNTAX IntegerOrString
SINGLE **VALUE**

security-accounting-identity Security-accounting-identity
:: = desd-att-accounting-identity

```
security-audit-identity Security-audit-identity
:: = desd-att-audit-identity

security-authentication-level Security-authentication-level
:: = desd-att-authentication-level

security-capability-type-1 Security-capability-type-1
:: = desd-att-capability-type-1

security-confidentiality-class Security-confidentiality-class
:: = desd-att-confidentiality-class

security-confidentiality-class-minimum Security-confidentiality-class-minimum
:: = desd-att-confidentiality-class-minimum

security-confidentiality-hierarchy Security-confidentiality-hierarchy
:: = desd-att-confidentiality-hierarchy

security-confidentiality-hierarchy-minimum Security-confidentiality-hierarchy-minimum
:: = desd-att-confidentiality-hierarchy-minimum

security-entity-identity Security-entity-identity
:: = desd-att-entity-identity

security-group Security-group
:: = desd-att-group

security-integrity-class Security-integrity-class
:: = desd-att-integrity-class

security-integrity-class-minimum Security-integrity-class-minimum
:: = desd-att-integrity-class-minimum

security-integrity-hierarchy Security-integrity-hierarchy
:: = desd-att-integrity-hierarchy

security-integrity-hierarchy-minimum Security-integrity-hierarchy-minimum
:: = desd-att-integrity-hierarchy-minimum

security-need-to-know Security-need-to-know
:: = desd-att-need-to-know

security-need-to-know-minimum Security-need-to-know-minimum
:: = desd-att-need-to-know-minimum

security-role Security-role
:: = desd-att-role

   END -- of Security Attribute Module
```

Appendix B
Mapping Services to Servers and End-System Components

This Appendix is not part of the Standard

In designing the abstract services of clause 2 it is necessary to see how these might be translated into real services and servers for which standard protocols will be needed, or how they might be translated into end-system or application process components for which an abstract service definition is sufficient to indicate the information each security service will require.

Abstract Security Services may be mapped onto a number of Security Service Applications. In practice more than one Abstract Service may be incorporated into a single Security Service Application. An example of this would be the combination of an Authentication Service and a Security Attribute Service into a single security service application. This approach allows both efficient, combined authentication/attribute issuing operations as well as separate authentication and attribute issuing operations where such is needed. Security Service Applications are real, possibly distributed applications, that can be named and registered for use in Open Systems. These Applications are of the same type as those of the Distributed Office Applications environment (ISO 10031); they may consist of one or more elements that are distributed over a number of physical systems. These elements are referred to as logical servers.

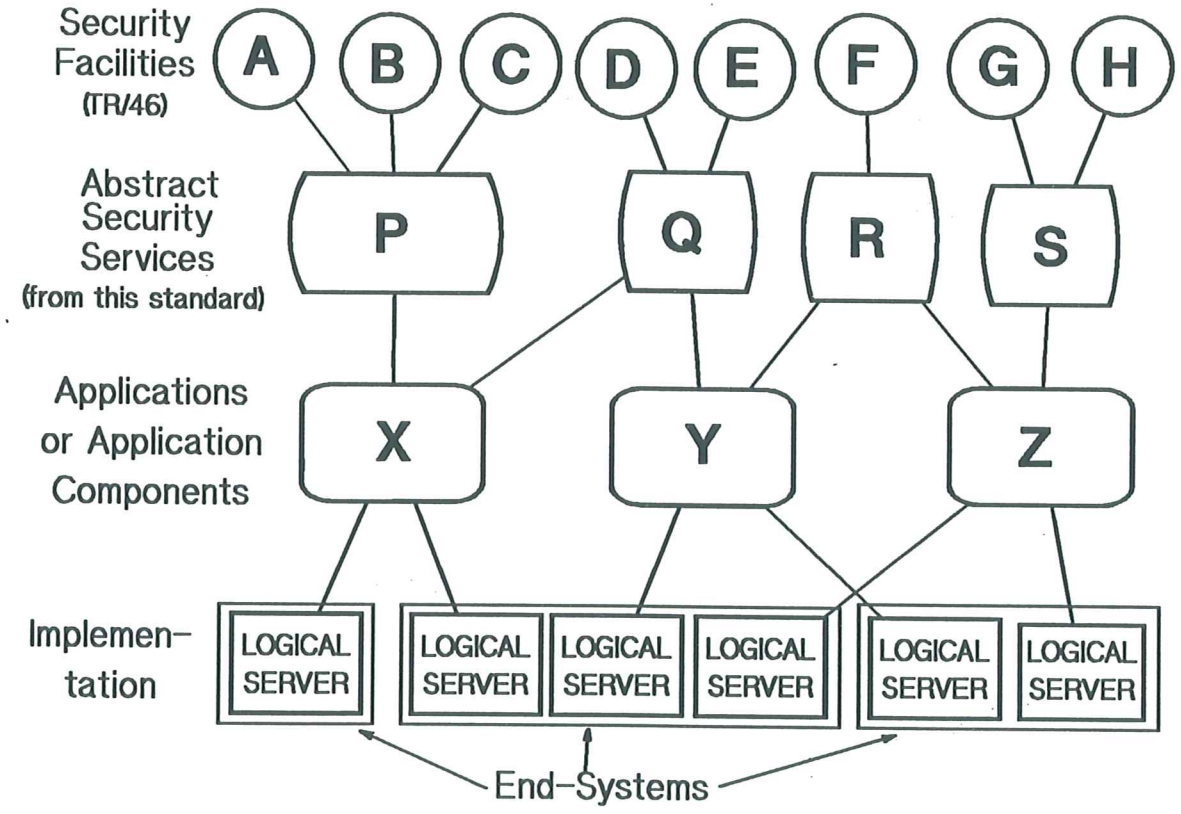

Figure B.1 - Mapping Security Facilities, Services and Applications to Implementation in End-Systems

The distributed elements of a Security Service Application are referred to here as logical (security) servers. A single end-system may support one or more logical servers of different types. The mapping of logical servers onto Application Entities is for further study. It should be noted that Abstract Service Definitions may be imported into other standards as a source of protocol elements in addition to the protocol elements specific for a given application or function. An example is the Document Filing and Retrieval Application Standard which could import the Abstract Service Definitions of a Secure Association Service, an Access

Authorisation Service and an Interdomain Service so as to incorporate protocol elements that support multi-domain access control.

An Abstract Security Service may be realised as an application component. For example an instance of "an Authorisation service" which is controlling access to the objects owned by a particular application may appear as a subsystem internal to the application, available only for the applications use. Thus the abstract concept of "an X Service" can appear in real distributed systems in multiple implementation instances and many forms, each implementation being separate from the others, all being related only in their support of a common policy.

B.1 Security Information Providing Services

There are three types of Security Information Providing Services: an Authentication Service, a Security Attribute Service and an Interdomain Service. A case can be made for combining these types into compound Security Service Applications with sets of service primitives that cover the sets of service primitives of the Security Services as defined in this document. Implementation of these combined service applications requires a distributed approach with access protocols and system protocols for interchange between each of the components. Some advantages of these combinations are:

- The authentication of a Principal and issuing a PAC can be combined in a single operation; this is more efficient than performing two separate operations.
- A single database that holds all security relevant information on the subjects of a given domain may be used to support all three types of service.

The two primary users of such combined service applications are the Subject Sponsor (when it logs-on human users or applications) and the Secure Association Service when it sets up connections across domain boundaries.

Any security information providing service may be mapped directly onto physical servers. Their function is to accept information, process it and return results; which is the classic action of a supportive service. Where access is made to a security information management server from a separate end-system a secure association must be used.

The services may be provided on end-systems themselves, by a network of distributed servers, or by a combination. In particular, a Security Attribute Service may be provided on individual end-systems where global system attributes need to be mapped onto end-system specific attributes.

B.2 Security Control Services and the Subject Sponsor

The two kinds of services concerned are an Authorisation Service and a Secure Association Service.

An Authorisation Service may be mapped onto physical servers as a single Security Service Application; However, there is a control function which has to be exercised at some point by a trusted component of the system. An Authorisation Service is often co-located with the objects to which it is authorising access. Where an Authorisation Service is placed on a separate end-system, then the calling end-system entity must be trusted to obey the authorisation decision.

A Secure Association Service is a function that needs to be provided in each of the two end-systems of an association. Although separate entities from a security point of view they may be considered components of each Application Context. Whether a Secure Association protocol is modelled as a separate ASE or whether it is considered a local function that interacts with the ACSE depends on design constraints outside the scope of this ECMA Standard which only defines the Data Elements needed.

A Subject Sponsor maps to a local function in each end-system which is directly accessible to human users or external applications that need a Subject Sponsor as entry point into the secure distributed system. From a systems security point of view the Subject Sponsor is a separate entity but from a protocol point of view it is part of each application.

B.3 Security Monitor Service and Recovery

An Audit Information Collection Service maps naturally to a single Security Service Application. The distributed elements of the service, such as sinks of information, may be mapped onto a number of physical end-systems. This would be particularly appropriate where reliability in this service is required. However, each of the other security services may implement local audit services, for their particular needs. A hierarchy of audit services may be used to avoid excessive amounts of information moving into a physically separate end-system.

Recovery is contained in each security service. Each service must provide a management port through which its internal recovery procedures may be invoked. Consequently this service is not mapped onto an end-system.

Appendix C
An Application, Application-Data Model

This Appendix is not part of the Standard

This model makes the distinction between applications and application data. It describes the functionality of a system in terms of three classes of components: Users, Applications and Application Data. In distributed systems communication protocols (e.g. OSI) support interaction between end-systems. A model of two end-systems and OSI communications is shown in figure C.1. This shows the possible interactions in a secure system, all of which pass through security services that constitute a secure environment for the objects. The only possible interactions are:

- User to Application (1)
- Application to Application (2)
- Application to Application Data (3)

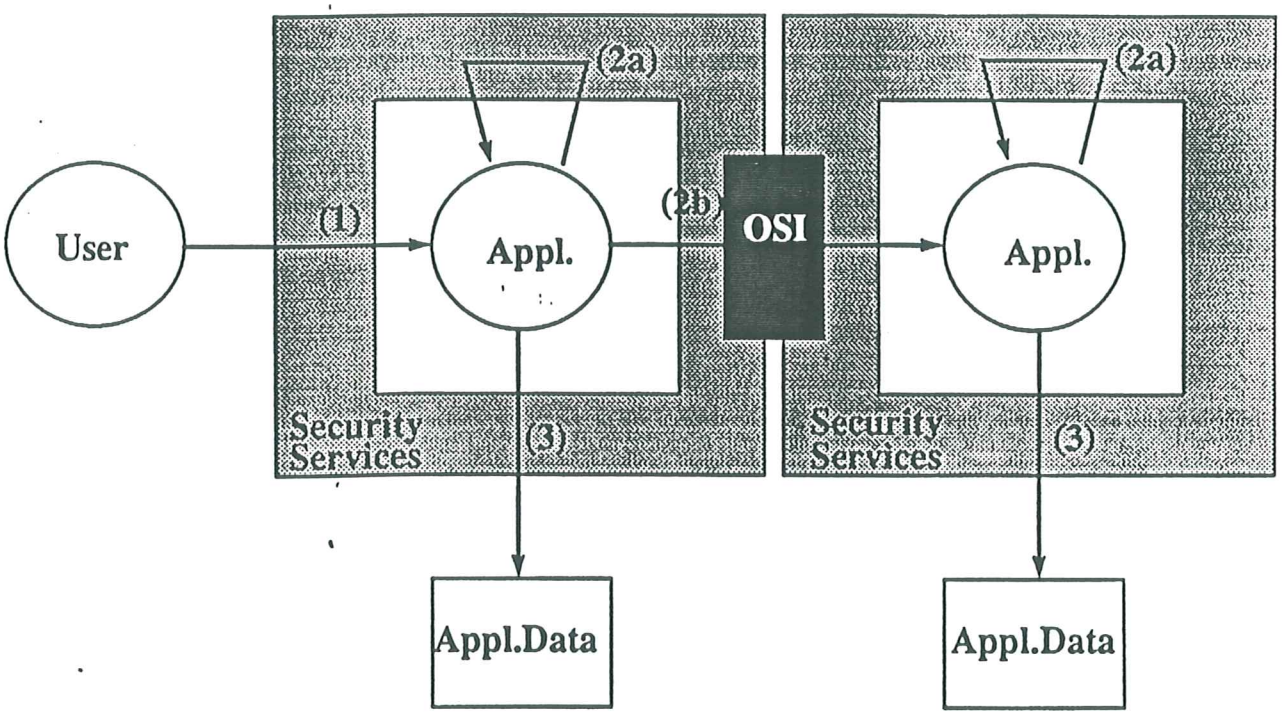

Figure C.1 - Interactions between Users, Applications, and Application Data

When a User interacts with the system it is through a Subject Sponsor. This is not a service, but security functionality which uses an Authentication Service and a Secure Association Service to control the interaction marked (1) in figure C.1. Applications interact with other Applications through a Secure Association Service. This service mediates and controls associations on the same end-system (2a), and between two end-systems via communication protocols (2b). The initiating Application does not distinguish between the two cases. A Secure Association Service needs to call an Authorisation Service which can authorise or refuse the interaction. This decision is based upon information provided by a Security Attribute Service. In case (2b) the target may belong to a different security domain making it necessary to use an Interdomain Service at the initiator and/or target side to map attributes. When Applications access their own Application Data (3) the Authorisation Service is used to authorise or refuse the access. This decision may be based upon privilege attributes of external users or applications on whose behalf the access is being requested. Finally there is a Security Audit Information Collection Service which is provided to support surveillance and maintenance of the security system. The above services are outlined in this ECMA Standard.

Appendix D
Interdomain Security

This Appendix is not part of the Standard

This appendix examines some of the aspects of using the Interdomain Service described in 2.2.1.3 and clause 8; Clause D.1 discusses ways of using the Interdomain Service in two extreme cases of trust and mistrust, and Clause D.2 discusses some aspects of interdomain data flow that are not covered by the Interdomain Service.

D.1 Use of the Interdomain Service under Different Policies

The following gives two examples of the use of the Interdomain Service under policies which are representative of two ends of a spectrum: mutually trusting domains and mutually distrusting domains. Within this spectrum various mixtures of trust and cooperation are possible but not discussed here.

D.1.1 Mutually Trusting Domains

D.1.1.1 Interdomain Policy

The Domain Administrators have agreed to recognise each other as trusted domains for a defined set of access privileges under the proviso that the Privilege Attributes associated with all interdomain access attempts must bear the seal of the Interdomain Authority of the initiating domain. The interworking policy is expressed as a set of object names and operations for each domain and the establishment of seal verification keys in each domain for the other domain. Establishing the relevant keys may require some communication between Interdomain Services in each domain.

D.1.1.2 Security Attribute Translation

The Domain Administrators may agree on a common interdomain syntax specific for their interchanges. Attribute syntax mapping may therefore occur both on exit of one domain and upon entry into the other domain.

D.1.1.3 Interdomain Access Operations

When an initiator in one domain wants to access an object in another domain, the privileges associated with the access request are sealed by its Interdomain Service and passed to the other domain where the request is re-sealed by the Authority Service there.

These access requests may vary from application bindings to specific operation requests and each will be authorised according to the Privilege Attributes that have resulted from the Interdomain Service translations.

D.1.1.4 Interdomain Service Involvement - initiating domain

Any request for interdomain access is routed to the Interdomain Service. The PAC associated with this request is translated into the agreed interdomain attribute syntax and sealed with the name of the Interdomain Authority agreed with the other domain. The request then is allowed to proceed, e.g. to set up a connection to an object in the other domain.

D.1.1.5 Interdomain Service Involvement - target domain

The PAC received with any request from the other domain is passed to the Interdomain Service. Here the policy constraints, with regard to the privileges agreed between the two domains, are verified. If the Privilege Attributes are authorised,

they are translated from the interdomain attribute syntax into the local attribute syntax, the PAC is sealed by the target Interdomain Service and the request is allowed to continue.

D.1.2 Mutually Distrusting Domains using an Intermediary

This situation, that of using a mutually trusted 3rd party (a broker), may also be appropriate where interaction between two domains does not justify a separate and direct 2-party agreement.

D.1.2.6 Interdomain Policy

Neither domain Authority will recognise requests for service from the other domain. However, each trust a third domain - a broker domain - to mediate each interchange. Both domains register with the broker domain those privileges that are allowed under their policies and they establish seal verification keys with the broker domain. Part of the agreement is that the broker domain will process only requests with privileges sealed by the appropriate Interdomain Service.

D.1.2.7 Attribute Syntax Translation

Security Administrations operating under a policy suggested here are unlikely to agree on a mutual syntax. Instead, their agreements with the broker domain are likely to stipulate the use of a standard attribute syntax as given in 3.7 of this ECMA Standard.

D.1.2.8 Interdomain Operations

When an Initiator in one domain wants to access an entity in the other domain, the privileges associated with an access request are sealed by the initiator Interdomain Service and passed to the broker domain where the privileges are validated and re-sealed by the broker's Interdomain Service before being passed to the target domain. The broker's Interdomain Service will verify that the privileges associated with the request fall under the agreed policy of the initiator and target domains. These access requests may vary from application bindings to specific operation requests.

D.1.2.9 Interdomain Service involvement - initiating domain

Privilege attributes associated with an interdomain request are translated into the standard attributed syntax and sealed with the name of the Interdomain Authority agreed with the broker domain. The request then is allowed to proceed.

D.1.2.10 Interdomain Service involvement - broker domain

All PACs received from the broker's client domains are verified against the applicable policy elements agreed withthe client's domain. The request is then allowed to proceed. This verification is done by the broker's Interdomain Service. Parameters supplied to the Service specify the client domains involved and possibly other parameters such as policy subsets. It is worth noting that the policy of the broker is limited to checking the PACs from a client against the policy agreed with the other.

D.1.2. 11 Interdomain Service involvement - target domain

The privileges received from the broker domain are routed to the Interdomain Service. Here the security attributes are translated from the standard syntax into the syntax of the target domain and the policy constraints, with regard to the privileges agreed with the broker domain, are verified. If they are authorised, the privileges associated with the request are sealed by the target Interdomain Service and the request is allowed to continue.

D.2 INTERDOMAIN DATA FLOW CONTROL

Some security policies for interdomain working may require that all data flow to and from objects outside a domain be controlled in real time. The Interdomain Service described in this ECMA Standard (clause 8) does not support this functionality. A policy can be enforced by the use of interdomain gateways through which all associations to objects outside the boundary must pass. This is shown in figure D.1.

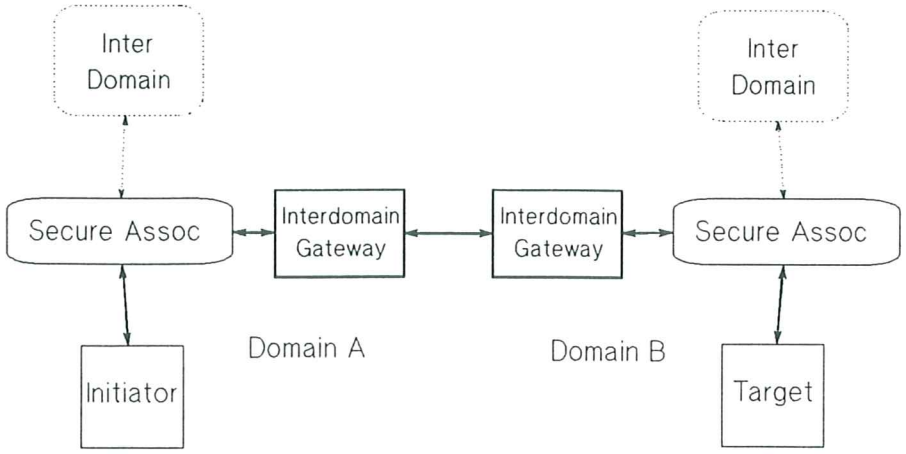

Figure D.1 - Position of Interdomain Gateway

Interdomain gateways would provide the following functionality:

- All associations can be authorised by the gateway; in this case the Authorisation Service used by the Secure Association Service of the initiating object does not have to contain any extra-domain rules; this Authorisation service may contain rules about which objects may have access to the Interdomain Gateway.
- All data exchanged with an object in a remote domain is scrutinised and controlled by the gateway.

To be able to monitor data across an association the Interdomain Gateway may need to know about its structure. In particular, to map embedded security information the gateway must be able to recognise such information (e.g. a security label) in the data stream.

Figure D.1 shows two interdomain gateways. Often only one of the interacting domains is concerned about interdomain data flow control, in which case only one gateway owned by that domain is required. Having two Gateways back-to-back owned by each domain must be a major impediment to efficient communications. Alternatively, it may be possible for a broker domain to support a single gateway, this will not be owned and operated by the domains that require the control.

Appendix E
Relationship between this Standard ECMA-138 and the Directory

This Appendix is not part of the Standard

The Directory [CCITT X.500, ISO 9594] provides a way to organise and store information related to named objects. This information is referred to as Attributes.

In general the Directory has no responsibility for the information placed in the Directory Information Base (DIB). The Directory is a repository for information and only assures the availability and consistency of the information in the DIB.

In addition the Directory provides digital signatures as a mechanism which allows Directory users to verify information integrity and origin. An Authority can place information in the DIB. When the Directory is used by someone to retrieve signed information the digital signature can be verified. If it is correct and the origin is a trusted Authority then the information itself can be trusted. To enable this the Public Key of the Authority must be known and trusted. Mechanisms like trust trees and cross certificates eliminates the need to know the Public Key of more than one Authority. For details see X.509.

This makes it possible to use the Directory as a tool in Security Management. For instance, linking Attributes such as Public Keys to named objects (human users, machines or applications). A method for doing this is described below:

An Authority places information which consists of a name associated with its Public Key in the Directory, typically a User Certificate. The Authority is for this reason called a Certification Authority (CA). The digital signature in the Certificate allows Directory users to trust Public Keys associated with named objects. Use of the Directory for Public Key Management is one specific but interesting application. If the Directory user is himself a human requesting the Certificate of another human, then both will trust the Public Keys of the other by inspection of the other party's Certificate. Thus enabling mutual authentication or Secret Key exchange by common trust in the Authority.

The main security feature of the Directory, in its present form, is thus to provide a secure mapping between objects and their public keys.

Although the purposes of this ECMA Standard and the Directory are different, there are several similarities. In particular this ECMA Standard supports a full range of security attributes associated in a secure manner with an object through the use of a PAC. The Directory recommendation only associates a single security attribute (a public key) in a secure manner with an object through the use of a User Certificate.

User Certificate	Privilege Attribute Certificate
Version	Syntax Version
Serial number	PAC Identifier
	Recovery Identifier
Issuer	Authority
Validity	Creation Time
	Validity Time
Subject	Privilege Attributes (may include Subject Identity)
Subject Public Key	(may be an attribute)
not included	Usage protection fields
not included	Audit identifier
not included	Charging identifier
Signature	part of the Seal
Signature Algorithm	part of the Seal

Table E.1 - Comparison of the Directory User Certificate and the Privilege Attribute Certificate

This ECMA Standard makes use of the same syntax as the Directory in defining security attributes. The Directory can therefore be used to store such security attributes. However, the Directory is not, in its present form, secure enough to do so. This ECMA Standard describes the data elements and abstract services with which the Directory could be enhanced. Provided the Directory, or parts of it, were made sufficiently secure, using the Directory to store attributes could be extended to support the storage of attributes for a Security Attribute Service.

The contents of the User Certificate of the Directory may be compared with the contents of the PAC defined in this Standard, this is summarised in table E.1. It is noted that all fields are covered with the addition of several other attributes protecting the PAC.

Appendix F
Transmission of Access Privileges

This Appendix is not part of the Standard

F.1 Introduction

The purpose of this Appendix is to establish requirements for the passing of privilege attributes between different components of a compound object in a multiple component proxy relationship. It will be seen that the full picture is complex, but that it can be greatly simplified in practice.

The Appendix is structured as follows:

- An initial assumption is described relating to the way in which entity privileges are modified according to their environment.
- The requirement is first illustrated by means of an example. This gives a view from a particular application object, of the sources and uses of the various privileges that might be required to execute operations on a second object. Three classes of operation are identified, each of which has different characteristics relating to privilege attribute handling.
- The example is extended in two stages to give a more general picture.
- Practical simplifications consistent with the description of privilege attribute handling given in Clause 10 are described.

F.2 Combination of Access Privileges

ECMA TR/46 describes how active entities in a system (human users and applications) acquire access privileges as a result of interactions with authentication and attribute facilities. These interactions are conducted by subject sponsor facilities.

Many security policies require that privileges, to be granted to an active entity, can be determined not only according to the entity's identity but also according to the environment from which it is accessing the system. In the ECMA model this is represented by a combination of what the underlying sponsor tells the authentication and attribute services, and the trust they have in the sponsor.

In principle, this trust can be represented by a set of privileges associated with the sponsor, paralleling the way in which privileges are associated with the sponsored entities themselves. Each of these components could acquire a PAC of its own which is presented whenever an access request is made. However, a simplifying assumption is made that the PAC granted to a user or application is *already* modified according to the security properties of the environment from which it is accessing the rest of the system. This modification may include restrictions or additions of privileges.

F.3 Example

The scenario analysed is that Object A1 under the control of user U is accessing Object A3 via Object A2. Each access takes the form of an *Operation* by A2 on A3. A3 may use the services of a further Object (A4) in the execution of the Operation; we use the term <MI>*Subcontracting* to describe this way of working. This is illustrated in the following page.

We now concentrate on the relationship between A2 and A3. We consider the Privileges that may need to be associated in some way with an Operation request by A2 on A3. There are three classes of privilege that may be required:

1. Privileges that A2 will need to obtain permission to perform the Operation.

2. Privileges that A3 itself may need if it is to Subcontract some of the Operation (in this case to A4).

3. Privileges that A2 may wish to establish (by means of this Operation) as a *Security Privilege Context* for future Operations by A2 on A3. We define the Security Privilege Context of A2 with respect to A3 (often abbreviated to "Context" in the rest of this Appendix) as privileges deposited with A3 which may be subsequently assumed by A2 for an identifiable set of Operations. The privileges apply for the identified Operations from the time of their being deposited until the Context is altered or terminated.

Each of these three classes of privileges may be obtained in principle from any of three different sources, though not all sources apply for all usages or all kinds of Operation:

1. A2's privileges already established, by a previous Operation, with A3 as being associated with the Context within which the current Operation is being performed.

2. Privileges passed by A2 to A3 as a parameter to the Operation. Note that these themselves may have been obtained by A2 from either or both of two sources; they may have been A2's own privileges or they may have been passed to A2 from A1 for A2's use.

3. Privileges belonging to A3 (apply only for the purpose of Subcontracting to A4).

We therefore have the following matrix for Operations by A2 on A3. In this matrix "n/a" means "not applicable".

	Privilege Context already established for A2 with A3		Passed as operation Parameter	
		Belonging to A3	A2's own	From A1
Needed by A2 to Perform Operation	A	n/a	B	C
Needed by A3 to Subcontract	n/a (note)	D	E	F
To establish new Privilege Context	n/a	n/a	G	H

Note F.1
We assume that Context privileges for A2 with A3 are not appropriate for use by A3 with A4.

If we say that each set of attributes is realised as a PAC, then in principle any or all of six PACs (containing attribute sets B, C, E, F, G and H) appear to need to be passed as parameters to an Operation. We shall see later that even with this number of PACs we have not yet established the full general case (for example A2 may obtain more than one PAC from A1), though in practice the real requirement can be simplified. We shall also show later that the sets A or D are not always relevant, though this does not directly affect requirements for the parameterisation of Operations.

We first distinguish between three different classes of Operation:

- Initial Connection
- Context Establishing Operations
- Other Operations

F.3.1 Initial Connection Operations

By definition we say that there is no previously established Context, so A does not apply. We also assume that A3 does not require any privileges from A2 in order to Subcontract an A2 to A3 connection; such Operations are not Subcontracted. This means that D, E and F are not required.

So we are left with B, C, G, and H. We shall later see that B and C can be brought together in the same parameter, and that G and H can similarly come together.

F.3.2 Context Establishing Operations

We define operations in this class as having the characteristic of being solely concerned with establishing or resetting Security Privilege Context within a connection that has already been established.

We first see that such an Operation's permissibility can be a function only of the current access environment. This is because the PAC which defines the new Context, in common with all PACs, is inherently self authorising. The mere ability to present a valid PAC means that the presenter possesses the privileges contained in it. It is only the current environment that might prevent its use (typically on the basis of being unable to withdraw extant privileges because of the current access situation, or on the basis that the current connection is not sufficiently secure to permit requested strong privileges).

This means that B and C are not required. By its nature, the Operation does not involve any Subcontracting so D, E, and F are not required.

So we are left with G and H; we shall see later that these can be brought together in one parameter.

F.3.3 Other Operations

This class relates to all other Operations not concerned with connection or establishing Context. These are the basic Operations which are the main purpose of the interaction between A2 and A3 (though it should be noted that practical implementations of real Operations can be compounds of Operations from more than one of the classes defined here).

By definition, Operations in this class do not modify Context, and G and H are not required. A and D may both apply. B, C, E, and F may all apply.

F.3.4 Summary

In summary, using the format of the matrix, we have the following PAC parameterisations that are relevant for the different Operation classes:

Initial Connection:	Context Establishment:	Other:
B C	- -	B C
- -	- -	E F
G H	G H	- -

Where:

B represents the initiating object's (A2's) own privileges,

C represents privileges obtained by the initiating object (A2) from the previous object (A1),

E is as B but for the target object (A3) to use in Subcontracting,

F is as C but for the target object (A3) to use in Subcontracting,
G is as B for establishing Context,
H is as C for establishing Context.

F.4 First Generalisation

We now expand on the contents of C, E and F.

First of all if we examine C (obtained by the initiator A2 from A1) we see that these privileges may be a mixture of A1's own privileges and privileges passed to A1 from user U.

We now turn the focus of attention from A2 to A3, and look at Operations that A3 may wish to execute on A4. A3's version of the C parameter may be a mixture of A3's own privileges and those passed to it from A2 (via a previous F parameter), which themselves may be mixed with the mixture previously described (this latter being obtained by A3 via an F parameter).

The situation is clearly recursive. If we sketch out the contents of the four parameters B, C, E and F in the box shape shown in the matrix we see the following most general case for a four object sequence:

U to A1

P(u,1)	–
P(u,2) P(u,3) P(u,4)	–

A1 to A2

P(1,2)	P(u,2)
P(1,3) P(1,4)	P(u,3) P(u,4)

A2 to A3

P(2,3)	P(1,3) P(u,3)
P(2,4)	P(1,4) P(u,4)

A3 to A4

P(3,4)	P(2,4) P(1,4) P(u,4)
–	–

Where P(i,j) is a set of privileges from original owner i for object j.

F.5 SECOND GENERALISATION

From the pattern developed above it can be seen that the general B, C, E, F matrix for an Operation by A(i) on A(i+1) in a series of N objects under control of a user U is:

	Passed as Operation Parameter			
	A(i)'s own	From A(i-1)		
Needed by A(i)	P(i,i+1)	P(u,i+1) P(1,i+1)	...	P(i-1,i+1)
Needed by A(i+1) for Subcontracting	P(i,i+2) ... P(i,N)	P(u,i+2) P(1,i+2)	...	P(i-1,i+2)
		P(u,i+3) P(1,i+3)	...	P(i-1,i+3)
			...	
			...	
		P(u,N) P(1,N)	...	P(i-1,N)

The general form gives a clue to an obvious simplification: there is no need to distinguish between the two columns in the above matrix, since the left hand column simply fits into the right hand column as an extra term. There is nothing special about A(i)'s privileges except that A(i) just happens to be the current initiator. We therefore get B combined with C, and E combined with F (and by the same token G combined with H) as follows:

	Passed as an Operation Parameter		
Needed by A(i)	P(u,i+1) P(1,i+1)	...	P(i-1,i+1) P(i,i+1)
Needed by A(i+1) for Subcontracting	P(u,i+2) P(1,i+2)	...	P(i-1,i+2) P(i,i+2)
	P(u,i+3) P(1,i+3)	...	P(i-1,i+3) P(i,i+3)
		...	
		...	
	P(u,N) P(1,N)	...	P(i-1,N) P(i,N)
To Establish new Context	X(u,i+1) X(1,i+1)	...	X(i-1,i+1) X(i,i+1)

Where the X's represent future contextual privileges and are therefore distinguished from the P values.

F.6 Practical Realisation

Parameterisation of the three classes of Operation can now be defined for the general case. Notice that when a target object subcontracts an Operation using privileges passed to it from an initiator, it is acting (at least in part) as the initiator's proxy; so in the ASN.1 given below the attributes concerned (the combined H/I parameter) have been called "proxy":

```
InitialConnectionOperation {
    -- other parameters,
    required SEQUENCE OF PrivilegeAttributeCertificate OPTIONAL,
    context SEQUENCE OF PrivilegeAttributeCertificate OPTIONAL}

SetContextOperation {
    -- other parameters,
    context SEQUENCE OF PrivilegeAttributeCertificate OPTIONAL}

OtherOperation {
    -- other parameters,
    required SEQUENCE OF PrivilegeAttributeCertificate OPTIONAL,
    proxy SEQUENCE OF PrivilegeAttributeCertificate OPTIONAL}
```

Clearly, support for the full generality of the model will be rare. Typically only two objects will be involved, the first being a User Sponsor whose own access privileges have been merged into the user's before the user's PAC is created.

In this simple case when the User Sponsor connects to the object the "required" parameter contains only the user's PAC, and the "context" parameter is null, defaulting to the value of the "required" parameter. Operations which set context will be parameterised with only one PAC from the user. Other Operations will only exceptionally require any additional privileges, which would appear, only when needed, as a single user-owned PAC in the "required" parameter.

Occasionally a third object may be subcontracted to. When this happens the connection to the third object would typically pass only the connecting object's PAC in the "required" parameter, with a null "context" parameter defaulting as before. The proxy parameter on the Operation request from the User Sponsor would contain at most only one PAC, though its source, its contents and its usage would vary according to policy. Under many policies no proxy PAC would be necessary.

Appendix G
Index

This index does not cover the Appendices