D

GSSKRB5	The IETF GSS Kerberos V5 definition which specifies details of the use of Kerberos V5 with GSS-API. It includes updates to RFC 1510 e.g. how to carry delegation information. It is specified in RFC 1964.
ECMAMECH	The ECMA GSS-API mechanism specified in ECMA-235. See also related standard ECMA-219 (Authentication and Privilege Attribute Security Application with related key distribution functions)
SESAMEMECH	The SESAME gss-api mechanism. This is a subset of the ECMA GSS Mechanism and is specified in draft-ietf-cat-sesamemech-00.txt.
SESAMEOV	The SESAME V4 Overview. This can be found via the web at www.esat.kuleuven.ac.be/cosic/sesame.html
SPKMMECH	The Simple Public-Key GSS-API Mechanism (SPKM). Internet Draft draft-ietf-cat-spkmgss-06.txt Jan. 1996.
X.509	ISO/IEC 9594-8, "Information Technology - Open Systems Interconnection - The Directory: Authentication Framework", CCITT/ITU Recommendation X.509, 1993.
DNstrings	IETF RFC 1779 A String Representation of Distinguished Names. March 1995.

References D

Note that these references are to definitions which are sometimes not in a single document.

OMG References

CORBA 2	The Common Object Request Broker: Architecture and Specification Revision 2.0 July 1995 plus some revisions to this as agreed by the interoperability revision task force for IOR tags.
CORBASEC	The CORBA Security specification. The current version of this at the time of writing this specification is OMG document 95-12-1. A revised version of that is imminent, so some of the revisions have been assumed in this specification - see B.3.
CSI RFP	Common Secure IIOP Request for Proposals (orb/96-01-03)

Security References

GSS-API	The Generic Security Services API as defined in IETF RFC 1508 (September 1993) and X/Open P308. An update to RFC 1508 has been produced by the ietf cat group. It is currently (at the time of writing this specification) draft-ietf-cat-gssv2-06.txt though has been approved to become an RFC.
XGGS-API	The extended gss-api supporting access control and delegation extensions defined in draft-ietf-cat-xgssapi-acc-cntrl-00.txt. This interface is also defined in the ECMA GSS-API Mechanism standard - ECMA-235
SNEGO	Simple negotiation GSS-API mechanism as defined in draft-ietf-cat-snego-02.txt.
KERBV5	The Kerberos V5 mechanism as defined in IETF RFC 1510 (September 1993).

This CSI specification is focused at inter-ORB interoperability, and therefore the IOR and SECIOP protocol. So it also does not specify the format of evidence tokens as they do not affect the SECIOP protocol. However, these evidence tokens may be passed between ORBs as parameters, and will not be understood by an ORB which does not use the same security technology.

In future, a mandatory interoperability evidence token format should be defined, at least for a limited number of types of evidence. This is expected to be compatible with the public key mechanism specified in this document and use X.509 version 3 certificates.

C.4 Audit Trail Interoperability

The CORBA Security specification includes an Audit Channel interface which allows applications and ORBs to write records to the audit trail. The way this Audit Service routes the audit records is not defined. This could be done using the OMG Event Service or other means. Also, the stored/on-the-wire format of audit records is not defined.

So there is no standard OMG defined method of bringing together audit records from different Audit Services.

C.2 Possible SECIOP Mechanism Enhancements

C.2.1 Mechanism and Option Negotiation

This specification assumes the mechanism identifiers in the IOR allow the client to choose what mechanisms and options to use when communicating with this target. Therefore, it does not define protocol exchanges to allow the client and target to negotiate either mechanisms or options.

However, if the target supports a number of mechanisms and options, the size of the IOR could become larger than desirable. So in future, it may be desirable to define protocol exchanges for mechanism negotiation, for example, using [SNEGO].

C.2.2 Further Key Distribution Options

The current CSI-ECMA protocol defines secret and public key options for key distribution and a hybrid option where secret keys are used within a domain, but public keys are used between domains. It does not define the protocol for use in the sort of hybrid system where the initiator uses secret key and target uses public key technology and vice versa.

This may be needed for interoperation between unlike domains. If so, further architectural options from ECMA 235 may need to be included in the specification.

3.2.3 Further Delegation Options at/above Level 2

The current level 2 specification supports restricting where an initiator's attributes can be used to targets identified by security name. Further options for restricting where a PAC may be delegated could be added. e.g. to restrict delegation to particular delegation policy domain. This would require definition of further "qualifier attributes" in the CSI-ECMA protocol (see application trust groups in ECMA 235). It would also require administration of this, which would best be done by extending the security policy administration in Chapter 6 of CORBASEC.

Composite delegation of the initiator plus immediate invoker kind is described in the CSI protocol, but is not mandatory at level 2. Further composite delegation options, including traced delegation, could be added.

C.3 Interoperability when using Non-Repudiation

The optional Non-repudiation service in the CORBA Security specification generates NR tokens. CORBASEC does not specify the technology used to generate these tokens or a standard form for them. Interoperability of evidence tokens would require a standard specification for such tokens.

Facilities not in this Specification C

C.1 Introduction

This specification includes interoperability between clients and target objects using the standard OMG interoperability protocol GIOP/IIOP.

It does not cover the following secure GIOP/IIOP interoperability cases which could be considered for future RFPs:

- interoperability utilising underlying secure communications for message integrity/confidentiality, but still using secure associations e.g. to delegate PACs
- negotiation of security mechanisms, or details associated with mechanisms (such as cryptographic profiles) as part of the protocol, not just in IORs
- security gateways/bridges (CSI security prevents use of other gateways of some types as described in B.2.3 above)

Also, if more advance facilities as listed in Appendix G of the CORBA Security specification are added to the specification, some of these have interoperability requirements which would require extensions to one or more of the protocols in this specification - see also B.3.1 above..

Also, this specification does not cover interoperability of information which is carried in parameters in object requests, rather than the SECIOP protocol itself. It therefore does not consider the following which could also be considered for future RFPs:

- interoperability of evidence tokens for non-repudiation generated using different security technology. These can be passed as parameters of requests.
- interoperability of audit trails generated using different technology

B.3.2 SECIOP Changes

Some minor changes to the SECIOP protocol in the 95-12-1 version of CORBASEC are assumed - see above. Also, a further change for a future CORBASEC revision has been identified as follows:

Multi-threading and Replay/Misordering Detection

As explained in 3.3.2, the replay and misordering detection facilities provided as part of the security mechanisms defined in this specification may not work in a multi-threaded environment where several threads use the same security association. This is because the calls on the security mechanism via GSS-API (or whatever) are not guaranteed to be in the same order that the messages are transmitted between the client and target object.

Replay and misordering detection should be provided as part of a common secure interoperability standard, but it is expected that the way to do this is to extend the SECIOP protocol. This is expected to require substantive changes to SECIOP which are not expected to be done during the initial clean up revision of the CORBA Security specification.

B.3.3 Positioning of Attribute Mapping

CORBASEC allows AccessPolicies to be replaced, so that one AccessPolicy can be used with different ORBs and often also with different operating systems. The value of the attributes transmitted between client and target by this interoperability specification are normally ORB and operating system independent. However, some Access Policies may wish to use attributes mapped/translated to more local ones, for example, operating system dependent ones.

This attribute mapping is currently assumed to be done automatically by the ORB (or Security Context object if the ORB conforms to the CORBASEC replacability option). In future, an extra replacability interface for attribute mapping may be provided as part of a CORBASEC revision to the replacability conformance option so that the mapping can be replaced independently of the security mechanisms and policies used.

- For signed or unprotected messages, the MessageInContext message is followed by the higher level protocol message being transmitted within a security context (i.e. GIOP message or message fragment). This specification assumes that the message_size field of MessageInContext includes the length of any such higher level protocol messages.
- The SECIOP DiscardContext message is assumed to include an optional discard_context_token. (The GSS_Delete_sec_context call returns a token and it is an aim of this specification to allow security implementations of Vault and SecurityContext objects as defined in CORBASEC to use GSS-API without knowledge of the underlying security mechanism used. Also, some protocols, including the CSI-ECMA one, protect this token to prevent some denial of service attacks).

Other implications on CORBASEC are described in the following sections.

B.3.1 IDL Implications

Delegation Interfaces

CSI level 2 supports controls on the delegation of credentials. The way of specifying these controls is not included in this, or the CORBASEC specification. It is assumed to be done by administrative action. For example, it may be done by associating the delegation controls with a user or an attribute set selected when the user logs on or selects attributes at other times. In line with CORBASEC, management of attributes associated with a principal is considered out-of-scope of this specification, but this should be reconsidered in future.

No facilities are currently defined for an application object to specify controls it wishes to apply on delegating its credentials. In future, such facilities may be considered for CORBASEC - see CORBASEC Appendix G section G.10.

Also, delegation policies defined in CORBASEC currently allow the administrator of the policy to specify only the basic delegation mode (no, simple or composite delegation), but not finer controls.

Values of Identities when mapped from Security Names

In the GSS Kerberos and SPKM protocols, the security name is the only identity of the principal transmitted to the target. As decribed in Chapter 2, this is then used for the values of the access_id and audit_id security attributes available to invocation access and audit policies and to applications. The form of this name is mechanism dependent. In future, a more mechanism independent form of name should be considered.

B.2.3 Interoperability Bridges

The secure interoperability standards defined here protect messages in transit for integrity and/or confidentiality.

CORBA 2 allows interoperability bridges which change the form of request for example, change to a different representation of data or make object references suitable for use outside this ORB.

This bridging may be done as part of the normal path of the ORB invocation, in which case, the message can be protected after the request has been transformed.

Some bridges are independent of the client and target ORBs, for example, may be in separate gateway systems. Any such bridges which are unaware of security will cease to work if the messages are protected. For such a bridge to work securely, it must act as a secure link in a delegation chain. No interfaces are defined in this specification to help this case as this is a research topic which should take into account other problems, for example, changing security technology in such gateways.

B.2.4 Encoding Rules

The SECIOP messages defined in CORBASEC and used here are specified in IDL and therefore encoded in CDR.

However, several of these messages include a security token which is defined as a sequence <octet> to the CORBA system. These tokens conform to the GSS-API token format and are designed so that an existing security implementation can be used. This also allows inter-operating between CORBA and other systems using the same security tokens. As such tokens are currently encoded in ASN.1, the tokens are defined in ASN.1, though they appear to CORBA as a sequence <octet>.

NB This is not a change to the CORBA Core.

B.3 CORBA Security

This CSI specification relies on the CORBA Security specification.

During the production of this specification, a number a changes to CORBASEC as defined in 95-12-1 have been identified. Some of these are editorial and are being included in the revised CORBASEC being produced by the clean up revision task force. The following bullets list the main revision items to CORBASEC assumed by this specification.

- AssociationOptions are assumed to be an unsigned short rather than a sequence of AssociationOption.

Changes to Existing Specifications B

B.1 Introduction

This includes changes to the CORBA Core and to the CORBA Security specifications.

B.2 CORBA Core Implications

B.2.1 Finding what Security is Supported

CORBASEC defined a `get_service_information` operation on the ORB.

For the CSI standard, extra ServiceInformation is returned when the ServiceType is Security. Two new Service Options are added:

```
const   ServiceOption      CommonInteroperabilityLevel0 = 10;
const   ServiceOption      CommonInteroperabilityLevel1 = 11;
const   ServiceOption      CommonInteroperabilityLevel2 = 12;
```

Also, the CSI specification defines values for the MechanismType returned when using mechanisms defined in this specification.

B.2.2 Use of Principal

The protocols defined here identify the initiating principal as part of the security tokens exchanged. They do not use the Principal `sequence<octet>` in the GIOP RequestHeader (version 1.0 or 1.1).

It is recommended that this field be removed to save space in messages, though this is not essential.

Also, use of the `get_principal` operation on the BOA (and the associated CORBA::Principal interface) should be deprecated, as it will not return information about the initiating principal as described in this specification. The CORBASEC `get_attributes` operation and `received_credentials` provide information about initiating principals.

Tag ids for the mechanisms are:

```
TAG_SPKM_1_SEC_MECH           = <value to be allocated by OMG>
TAG_SPKM_2_SEC_MECH           = <value to be allocated by OMG>
TAG_KerberosV5_SEC_MECH       = <value to be allocated by OMG>
TAG_CSI_ECMA_Secret_SEC_MECH  = <value to be allocated by OMG>
TAG_CSI_ECMA_Hybrid_SEC_MECH  = <value to be allocated by OMG>
TAG_CSI_ECMA_Public_SEC_MECH  = <value to be allocated by OMG>
```

Each protocol supports a number of cryptographic profiles. These are defined as:

```
typedef unsigned short    CryptographicProfile;
```

Five cryptographic profiles are defined for the SPKM protocol:

```
const   CryptographicProfile   MD5_RSA = 20;
const   CryptographicProfile   MD5_DEC_CBC = 21;
const   CryptographicProfile   DES_CBC = 22;
const   CryptographicProfile   MD5_DES_CBC_SOURCE = 23;
const   CryptographicProfile   DES_CBC_SOURCE = 24;
```

Four cryptographic profiles are defined for the GSS Kerberos protocol:

```
const   CryptographicProfile   DES_CBC_DES_MAC = 10;
const   CryptographicProfile   DES_CBC_MD5 = 11;
const   CryptographicProfile   DES_MAC = 12;
const   CryptographicProfile   MD5 = 13;
```

Four cryptographic profiles are defined for the CSI-ECMA protocol:

```
const   CryptographicProfile   FullSecurity = 1;
const   CryptographicProfile   NoDataConfidentiality = 2;
const   CryptographicProfile   LowGradeConfidentiality = 3;
const   CryptographicProfile   AgreedDefault = 5;
```

The MechanismType used in IDL calls in CORBASEC operations is a string. For CSI mechanisms, this string is the mechanism id followed by zero, one or more cryptographic profiles separated by commas.

The mechanism id the string form of the integer tag value of the appropriate TAG_x_SEC_MECH (see tag ids above). Each cryptographic profile is represented as the string form of the CryptographicProfile value (see cryptographic profiles above).

A.3 Protocol Definitions

This specification defines the details of the security tokens in in SECIOP messages for all the CSI mechanisms. This is often done by reference to other specifications, rather than by full definition in this document. It is therefore not appropriate to give a full specification of these protocols in this appendix.

IDL and Protocol Summary A

A.1 Introduction

The common interoperability definition is split as follows:

- IDL for the new IOR tags defined in this document and the representation of the CSI security mechanisms in the MechanismType in CORBA Security IDL..
- ASN.1 for the security tokens which appear in the SECIOP protocol definition in CORBASEC as sequence <octet>.

This specification relies on datatypes defined in CORBASEC such as Security Attributes, MechanismType, Association Options and the SECIOP protocol definition. Note, however, that some changes to these are required such as some new values for existing CORBA or CORBASEC IDL - see Appendix B.

A.2 IDL Summary

The TAG_x_SEC_MECH tags for all the mechanisms defined here have the same form as shown below:

```
struct <mechanism name>   {
    AssociationOptions              target_supports;
    AssociationOptions              target_requires;
    sequence<CryptographicProfile>  crypto_profiles;
    sequence <octet>                security_name
};
```

Where <mechanism name> is replaced by one of the following values:

```
SPKM_1
SPKM_2
KerberosV5
CSI_ECMA_Secret
CSI_ECMA_Hybrid
CSI_ECMA_Public
```

}

integKeySeed
A random number, optionally concatenated with a time value to ensure uniqueness, used as input to the one way function specified in integKeyDerivationInfo.

confKeySeed
A random number, optionally concatenated with a time value to ensure uniqueness, used as input to the one way function specified in confKeyDerivationInfo.

integKeyDerivationInfo
Key derivation information for the integrity dialogue key, as follows:

> **owfId**
> The one way algorithm which takes the basic key XOR the seed as input, resulting in the integrity dialogue key.
>
> **keySize**
> The size of the key in bits. If the algorithm identified by owfId produces a larger key, it is reduced by masking to this length, losing its most significant end.

confKeyDerivationInfo
Key derivation information for the confidentiality dialogue key. The fields in this construct have the same meanings as defined above for the integrity dialogue key.

Note:
It may be insecure to specify the same derivation algorithms and seeds for both integrity and confidentiality dialogue keys, particularly if they are to be of different lengths.

integDKuseInfo
Information describing how the integrity dialogue key is to be used, as follows:

> **useAlgId**
> The secret or public reversible encryption algorithm with which the integrity dialogue key is to be used.
>
> **useHashAlgId**
> The one way function with which the integrity dialogue key is to be used. It is the hash produced by this algorithm on the data to be protected which is encrypted using useAlgId.

confDKuseInfo
Information describing how the confidentiality key is to be used. The useHashAlgId construct is not used here.

Field	Value/Constraint
-- options	not used - all bits set to zero
-- conf_alg	not used - use NULL CHOICE
-- intg_alg	not used - use a SEQUENCE OF with zero elements
- validity	mandatory
- key_estb_set	only one element supplied containing gss-key-estb-alg
- key_estb_req	contains KeyEstablishmentData with targetApplication field missing
- key_src_bind	missing
req_integrity	sig_integ mandatory
certif_data	only userCertificate field supported
auth_data	missing

Definitions of KeyEstablishmentData and gss-key-estb-alg are given in 6.8.4 above.

6.9 Dialogue Key Block

Dialogue Key Block constructs are used to specify how the integrity dialogue key and confidentiality dialogue key should be derived from the basic key, and specify the cryptographic algorithms with which the keys should be used. Dialogue keys are explained above. The syntax is as follows:

```
 DialogueKeyBlock   ::=   SEQUENCE {
    integKeySeed           [0] SeedValue,
    confKeySeed            [1] SeedValue,
    integKeyDerivationInfo [2] KeyDerivationInfo   OPTIONAL,
    confKeyDerivationInfo  [3] KeyDerivationInfo   OPTIONAL,
    integDKuseInfo         [4] DKuseInfo           OPTIONAL,
    confDKuseInfo          [5] DKuseInfo           OPTIONAL
}
SeedValue  ::= SEQUENCE {
    timeStamp     [0]   UTCTime                    OPTIONAL,
    random        [1]   BIT STRING
}
KeyDerivationInfo::= SEQUENCE {
    owfId         [0]   AlgorithmIdentifier,
    keySize       [1]   INTEGER
}
DKuseInfo    ::= SEQUENCE {
    useAlgId        [0]   AlgorithmIdentifier,
    useHashAlgId    [1]   AlgorithmIdentifier    OPTIONAL
```

Field	Value/Constraint
-- creationTime	creation time of publicKeyBlock
- signature	contains all the signing information as well as the actual signature bits
- certificate	optional

6.8.5 CSI-ECMA Public Mechanism

In this scheme, both client and target posses a private/public key pair and neither use a KDS.

To establish the client-target association, the client constructs a targetKeyBlock containing a basic key encrypted under the target's public key. The target key block is signed with the client's private key. On receipt of the targetKeyBlock, the target directly establishes a basic key from it.

The *asymmetric* key distribution scheme:

- has a mechanism id of CSI_ECMA_Public.
- uses an SPKM_REQ in the targetKeyBlock of the initial_context_token.

This mechanism has only a profile of the SPKM_REQ as defined below.

Profile of SPKM_REQ used in Public Key Mechanism

The following table indicates which optional fields must be present in the SPKM_REQ in the targetKeyBlock for the CSI_ECMA_Public mechanism and indicates the values which are required to be present in all fields.

Field	Value/Constraint
requestToken	
- tok_id	not used - fixed value of '0'
- context_id	not used - fixed value of bit string containing one zero bit
- pvno	not used - fixed value of bit string containing one zero bit
- timestamp	creation time of SPKM_REQ - required
- randSrc	random bit string
- targ_name	X.500 Name of target AEF
- src_name	X.500 Name of initiator
- req_data	
-- channelId	not used - octet string of length one value '00'H
-- seq_number	missing

Profile of Ticket as used in hybridInterdomain scheme

Note that the krb5Ticket part of this is identical to that used in the CSI_ECMA_Secret key mechanism except that the EncTicketPart is encrypted with the temporary key used between KDSs rather than the target's key.

Field	Value/Constraint
krb5Ticket	
- tkt-vno	5
- realm	initiator domain name in Kerberos realm name form
- sname	target application name including the realmof the target
-- EncTicketPart	encrypted with temporary key (which is in turn encrypted within the keyEstablishmentData field)
--- flags	only bits 6, 10 and 11 can be meaningful in the context of the CSI-ECMA protocol, the rest are ignored
--- key	the basic key
--- crealm	initiator domain name in Kerberos realm name form
--- cname	principal name of the initiator (in the case of delegation the cname will be that of the delegate)
--- transited	not used
--- authtime	the time at which the initiator was authenticated
--- starttime	not used
--- endtime	the time at which the ticket becomes invalid
--- renew-till	not used
--- caddr	not used
--- authorization-data	contains the PPID corresponding to cname
publicKeyBlock	
- signedPKBPart	
-- encryptedKey	KeyEstablishmentData structure
-- encryptionMethod	gss-key-estb-alg
-- issuingKDS	X.500 name of initiator's KDS (the signer)
-- uniqueNumber	creation time of publicKeyBlock plus a random bit string
-- validityTime	only one period allowed

targetName
If present, contains the name of the target application. This is necessary for some of the KD-schemes.

nameHashingAlg
Specifies the algorithm which is used to calculate the hashedName field of the PlainKey.

hniPlainKey
hniIssuingKDS
Used as input to a hashing algorithm as a general means to prevent ciphertext stealing attacks.

plainKey
Contains the actual bits of the plaintext key which is to be established.

hashedName
A hash of the name of the encrypting KDS calculated using the plainkey and KDS name as input (within the HashedNameInput structure). The algorithm identified in nameHashingAlg is used to calculate this value.

targetName
If present, contains the name of the target for which the PublicTicket was originally produced. This may be different from the targetIdentity field of the initialContextToken if caching of PublicTickets has been implemented.

Key Establishment Algorithm

The PublicKeyBlock in this mechanism and the SPKM_REQ construct used in scheme 6 requires a sequence of key establishment algorithm identifier values to be inserted into the key_estb_set field. The OBJECT IDENTIFIER below is defined as the (single) key establishment "algorithm" for ECMA mechanisms:

```
gss-key-estb-alg AlgorithmIdentifier ::= {kd-schemes, NULL }
```

gss-key-estb-alg
This AlgorithmIdentifier identifies the key establishment algorithm value to be used within the key_estb_set field of an SPKM_REQ data element as the one defined by ECMA.

This algorithm is used to establish a symmetric key for use by both the initiator and the target AEF as part of the context establishment. The corresponding key_estb_req field of the SPKM_REQ will be a BIT STRING the content of which is a DER encoding of the KeyEstablishmentData element.

krb5Ticket
The Kerberos Ticket which contains the basic key. The encrypted part of this ticket is encrypted using the key found within the encryptedPlainKey field of the KeyEstablishmentData in the PublicKeyBlock.

publicKeyBlock
Contains the key used to protect the krb5Ticket encrypted using the public key of the recipient and signed by the encryptor (i.e. the context initiator's KD-Server).

signedPKBPart
The part of the publicKeyBlock which is signed. The **keyEstablishmentData** field contains the KeyEstablishmentData (defined below), i.e. the actual encrypted temporary key. The **encryptionMethod** indicates the algorithm used to encrypt the encryptedKey. The **issuingKDS** is the name of the KD-Server which produced the PublicTicket. The **uniqueNumber** is a value (containing a timestamp and a random number) which prevents replay of the PublicTicket. **validityTime** specifies the times for which the PublicTicket is valid. creationTime contains the time at which the PublicTicket was created.

signature
Contains the signature calculated by the issuingKDS on the signedPKBPart field.

certificate
If present, contains the public key certificate of the issuing KDS.

Key establishment data elements

These are used in public key establishment mechanisms.

```
KeyEstablishmentData ::= SEQUENCE {
   encryptedPlainKey [0]   BIT STRING,-- encrypted PlainKey
   targetName        [1]   Identifier   OPTIONAL,
   nameHashingAlg    [2]   AlgorithmIdentifier   OPTIONAL
}

 HashedNameInput ::= SEQUENCE {
   hniPlainKey       [0]   BIT STRING,-- same as plainKey
   hniIssuingKDS     [1]   Identifier
}

PlainKey ::= SEQUENCE {
   plainKey         [0]   BIT STRING,  -- The cleartext key
   hashedName       [1]   BIT STRING
}
```

encryptedPlainKey
Contains the encrypted key. The BIT STRING contains the result of encrypting a PlainKey structure.

6.8.4 CSI-ECMA Hybrid Mechanism

In this scheme, the initiator shares a secret key with its KDS and the target shares a secret key with its KDS (which is different). In addition, each KDS possesses a private/public key pair.

To establish the client-target association, the client gets a targetKeyBlock from its KDS containing the basic key encrypted under a temporary key and the temporary key encrypted under the target's KDS's public key. The targetKeyBlock is also signed using the initiator KDS's private key.

On receipt of the targetKeyBlock, the target transmits it to its KDS and gets back the basic key encrypted under the long term secret key it shares with its KDS.

The *hybridInterdomain* key distribution scheme:

- has a mechanism id of CSI_ECMA_Hybrid in the IOR.
- uses a Public ticket in the targetKeyBlock of the initial_context_token, as described below.
  A modified Kerberos TGS can be used as the KDS in this case.

Hybrid inter-domain key distribution scheme data elements

```
PublicTicket ::= SEQUENCE{
  krb5Ticket     [0]   Ticket,
  publicKeyBlock [1]   PublicKeyBlock}

PublicKeyBlock ::= SEQUENCE{
  signedPKBPart  [0]   SignedPKBPart,
  signature      [1]   Signature OPTIONAL,
  certificate    [2]   Certificate OPTIONAL
}
 SignedPKBPart ::= SEQUENCE{
  keyEstablishmentData [0]   KeyEstablishmentData,
  encryptionMethod     [1]   AlgorithmIdentifier   OPTIONAL,
  issuingKDS           [2]   Identifier,
  uniqueNumber         [3]   UniqueNumber,
  validityTime         [4]   TimePeriods,
  creationTime         [5]   UTCTime
}

UniqueNumber ::= SEQUENCE{
  timeStamp      [0] UTCTime,
  random         [1] BIT STRING
}
```

Profile of Ticket as used in symmIntradomain scheme

The following table indicates which optional fields must be present in the Kerberos ticket for the CSI_ECMA_Secret mechanism and indicates the values which are required to be present in all fields.

Field	Value/Constraint
tkt-vno	5
realm	ticket issuer's domain name in Kerberos realm name form
sname	target application name including the realm of the target
- EncTicketPart	encrypted with long term key of target AEF
-- flags	only bits 6, 10 and 11 can be meaningful in the context of the CSI-ECMA protocol, the rest are ignored
-- key	the basic key
-- crealm	initiator domain name in Kerberos realm name form
-- cname	principal name of the initiator (in the case of delegation the cname will be that of the delegate)
-- transited	not used
-- authtime	the time at which the initiator was authenticated
-- starttime	not used
-- endtime	the time at which the ticket becomes invalid
-- renew-till	not used
-- caddr	not used
-- authorization-data	contains the PPID corresponding to cname

The Kerberos Ticket's authorization_data field contains the PPID of the context initiator, as formally defined below.

```
ECMA-AUTHORISATION-DATA-TYPE ::= INTEGER { ECMA-ADATA (65) }
ECMA-AUTHORISATION-DATA ::= SEQUENCE {
  ecma-ad-type   [0] ENUMERATED  {ppidType  (0)},
  ecma-ad-value  [1] CHOICE  {ppidValue [0]SecurityAttribute}}
```

ppidType
Indicates the type of the authorisation data which in included in the Ticket.

ppidValue
This value is used in the ppQualification PAC protection method as defined above.

- hybridInterdomain: In this case, the targetPart field is not supplied. The PublicTicket contains a Kerberos ticket.
- asymmetric: the targetKDSpart is not supplied and the targetPart contains an SPKM_REQ.

The following table shows the different syntaxes used for targetKDSpart and targetPart for the defined KD-schemes. "Missing" in the tables means that the relevant construct is not supplied.

KD-Scheme name	kdSchemeOID	targetKDSpart	targetPart
symmIntradomain	{kd-schemes 1}	Missing	Ticket
hybridInterdomain	{kd-schemes 3}	PublicTicket	Missing
asymmetric	{kd-schemes 6}	Missing	SPKM_REQ

Further options are possible in future by defining further kd-schemes. For example, ECMA 235 also defines options for:

- initiators with public keys and targets with secret keys
- initiators with secret keys and targets with public keys

6.8.3 CSI-ECMA Secret Key Mechanism

In this scheme, the client and target each share different secret keys with the same Key Distribution Server.

To establish the association, between the client and target, the client obtains a targetKeyBlock from its KDS containing a basic key encrypted under the target's long term key. On receipt of the targetKeyBlock, the target can extract the basic key from it.

The *symmIntradomain* key distribution scheme:

- has a mechanism id of CSI_ECMA_Secret.
- uses a Kerberos ticket in the targetKeyBlock of the initial_context_token. An unmodified Kerberos TGS can be used as the KDS in this case.

The form of this information depends on the key distribution configuration in place.

6.8.1 Keying Information Syntax

```
TargetKeyBlock ::= SEQUENCE {
    kdSchemeOID       [2]   OBJECT IDENTIFIER,
    targetKDSpart     [3]  ANY         OPTIONAL,
                           -- depending on kdSchemeOID
    targetPart        [4]  ANY          OPTIONAL
                           -- depending on kdSchemeOID
}
```

kdSchemeOID
Identifies the key distribution scheme used. Allows the targetAEF to determine rapidly whether or not the scheme is supported. It also allows for the easy addition of future schemes.

targetKDSpart
Part of the Target Key Block which is processable only by the KDS of the target AEF. This part is sent by the target AEF to its local KDS, in order to get the basic key which is in it. It must always contain the name of a target "served" by the targetAEF in question. The mapping between the name of the application and the name of the target AEF is known to the target AEF's KDS which is able to authenticate which target AEF is issuing the request for translating the targetKDSpart. It can then verify that the AEF is one which is responsible for the application name contained in the targetKDSpart. If it is, the key is released and is sent protected back to the requesting AEF. targetKDSpart should include data that enables the KDS of the target AEF to authenticate the KDS of the initiator. When the "Primary Principal Qualification" protection method needs to be used for the PAC, unless there is an accompanying targetPart, targetKDSpart must contain the appropriate primary principal security attributes (which is always true in this specification).

targetPart
Part of the Target Key Block which is processed only by the target AEF. When there is no targetKDSpart it is processable directly; otherwise it can only be processed after the target KDSpart has been processed by the KDS of the target AEF, and the appropriate Keying Information has been returned to the AEF. The targetPart construct should include data that enables the target AEF to authenticate the KDS of the initiator. When the "Primary Principal Qualification" protection method needs to be used for the PAC, targetPart must contain the primary principal security attributes.

6.8.2 Summary of Key Distribution Schemes

This specification defines three key distribution schemes. These are:

- symmIntradomain: using a secret key technology within a domain. In this case, the targetKDSpart of the TargetKeyBlock is not supplied and the targetPart contains a Kerberos ticket.

```
    secretAlgId        [1]   AlgorithmIdentifier    OPTIONAL,
    hashAlgId          [2]   AlgorithmIdentifier   OPTIONAL,
    targetName         [3]   Identifier           OPTIONAL,
    keyId              [4]   INTEGER              OPTIONAL
}
```

sealValue
The value of the seal. It is the result of a secret encryption of a hash value of a set of octets (which are the DER encoding of some ASN.1 type)

secretAlgId
An optional indicator of the sealing algorithm.

hashAlgId
Only present if the secretAlgId does not specify which hashing algorithm is used.

targetName
This field identifies the targetAEF or target with which the secret key used for the seal is shared

keyId
This serial number together with the targetName uniquely identifies the secret key used in the seal.

6.8 Basic Key Distribution

The TargetKeyBlock is structured as follows:

- an identifier (kdSchemeOID) for the key distribution scheme being used, which takes the form of an OBJECT IDENTIFIER,
- a part which, if present, the target AEF needs to pass on to its KDS (targetKDSPart - will be present only when the target AEF's KDS is different from the initiator's),
- a part which, if present, can be used directly by the target AEF (targetPart).

When a targetAEF using a separate KDS receives the targetKeyBlock, it first checks whether it supports the key distribution scheme indicated in kdsSchemeOID. Two different cases need to be considered:

1 Only the targetPart is present. The target AEF computes the basic key directly, using the information present in the TargetPart. The syntax of targetPart is scheme dependent. Expiry information can optionally be present in targetPart. If supported by the scheme, the Primary Principal attributes of the initiator will also be present for PAC protection under the Primary Principal Qualification method (see above).

2. Only the targetKDSPart is present. The targetAEF forwards the TargetKeyBlock to its KDS. In return it receives a scheme dependent data structure which allows the target AEF to determine the basic key and, if supported by the scheme, the Primary Principal attributes of the initiator for PAC protection purposes. Expiry information can optionally be present in the targetKDSPart.

A signature may be accompanied by information identifying the Certification Authority under which the signature can be verified, and with an optional convenient reference to or the actual value of the user certificate for the private key that the signing authority used to sign the certificate.

```
CheckValue ::= CHOICE{
    signature      [0]   Signature
    -- only signature supported here
}
Signature ::= SEQUENCE{
    signatureValue        [0]   BIT STRING,
    publicAlgId           [1]  AlgorithmIdentifier   OPTIONAL,
    hashAlgId             [2]  AlgorithmIdentifier   OPTIONAL,
    issuerCAName          [3]   Identifier    OPTIONAL,
    caCertInformation     [4]   CHOICE {
           caCertSerialNumber    [0]    INTEGER,
           certificationPath     [1]    CertificationPath
    }
    OPTIONAL
}
--CertificationPath is imported from [ISO/IEC 9594-8]
```

signatureValue
The value of the signature. It is the result of an public encryption of a hash value of the certificateBody.

publicAlgId
Only present if the certificate body is encrypted, then it is a duplication of the algId value in "commonContents". This is not needed in CSI-ECMA.

hashAlgId
Only present if the certificate body is encrypted, then it is a duplication of the hashAlgId value in "commonContents". This is not needed in CSI-ECMA.

issuerCAName
The identity of the Certification Authority that has signed the user certificate corresponding to the private key used to sign this certificate.

caCertInformation
Contains either just a certificate serial number which together with the issuerCAName uniquely identifies the user certificate corresponding to the private key used to sign this certificate, or a full specification of a certification path via which the validity of the signature can be verified. The latter option follows the approach used in [ISO/IEC 9594-8].

The Seal structure is used in the Tokens defined above.

```
Seal ::= SEQUENCE{
    sealValue          [0]   BIT STRING,
```

For the Target Qualification protection method, the MethodId is `targetQualification` and the syntax for Mparms is `securityAttribute`.

For the Delegate/Target Qualification protection method, the MethodId is delegate`targetQualification` and the syntax for Mparms is `securityAttribute`.

The security attribute in the target and delegate/target protection method is a qualifier attribute as defined in 6.6.4.

External Control Values Construct

When using the controlProtectionValues method a PAC protected under that method may be accompanied by one or more control values and indices to the method occurrences in the certificate to which they apply. Also, when such a certificate is being issued to a requesting client, the CV values it will need in order to use that certificate may need to be returned with it.

```
ECV ::= SEQUENCE {
    crypAlgIdentifier  [0]   AlgorithmIdentifier     OPTIONAL,
    cValues            [1]   CHOICE {
                               encryptedCvalueList [0]BITSTRING,
                               individualCvalues  [1]CValues
                               }
}
CValues ::= SEQUENCE OF SEQUENCE {
    index          [0]   INTEGER,
    value          [1]   BIT STRING
}
```

crypAlgIdentifier
This specifies the encryption algorithm of the control values.

cValues
An ECV construct can contain either an encrypted list of control values in the encryptedCvalueList field, or a list of individually control values in individualCvalues.

If the encryptedCvalueList choice is made, the whole list is encrypted in bulk, but the in-clear contents of this field are expected to have the syntax CValues. If the individualCvalues choice is made, values are individually encrypted in the value fields of the list. Encryption is always done under the basic key protecting the operation.

In the case of the controlProtectionValues method, value is a CV, and index is then the index of the method occurrence in the certificate, starting at 1.

6.7.3 Check value

In this specification a PAC is protected by being digitally signed by the issuer.

Protection Methods

A method consists of a method id and parameters (methodParams). The method id determines the syntax for the type of methodParams.

```
MethodGroup ::= SEQUENCE OF Method
Method ::= SEQUENCE{
    methodId         [0] MethodId,
    methodParams     [1] SEQUENCE OF Mparm    OPTIONAL
}
MethodId ::= CHOICE{
    predefinedMethod [0] ENUMERATED {
                          controlProtectionValues  (1),
                          ppQualification  (2),
                          targetQualification   (3),
                          delegateTargetQualification (4)
                          }
}
Mparm ::= CHOICE{
    pValue               [0]   PValue,
    securityAttribute    [1]   SecurityAttribute
}
PValue ::= SEQUENCE{
    pv                   [0]   BIT STRING
    algorithmIdentifier    [1]   AlgorithmIdentifier    OPTIONAL
}
CertandECV ::=   SEQUENCE {
    certificate      [0]   GeneralisedCertificate,
    ecv              [1]   ECV,             OPTIONAL}             -
    - ECV is defined in later
```

methodId
Identifies a protection method. Methods can be used in any combination, and except where stated otherwise, multiple occurrences of the same method are permitted. The choice of methodId determines the permitted choices of method parameters in the methodParams construct as described below.

methodParams
Parameters for a protection method. The semantics of each protection method is described in section 5.2 above.

For the Primary Principal Qualification Method, the MethodId is `ppQualification` and the syntax of Mparm is `securityAttribute`. Its value is defined in 6.2.8 above.

For the PV/CV method, the MethodId is:`controlProtectionValues` and the syntax of Mparm is: `pValue`.

```
                    },
                    -- the actual restriction in a form undefined here
    type        [2] ENUMERATED  {
                        mandatory (1),
                        optional    (2)}      DEFAULT mandatory,
    targets     [3] SEQUENCE OF SecurityAttribute    OPTIONAL
}                   -- applies to all targets if this is omitted
```

pacSyntaxVersion
Syntax version of the PAC.

protectionMethods
A sequence of optional groups of Method fields used to protect the certificate from being stolen or misused. For a full description see below.

pacType
Indicates whether the privileges contained in the PAC are those of a Primary Principal (e.g. the client), or of a Secondary Principal (e.g. the user). In this specification, it is always a PAC of a secondary principal untempered by the privileges of a Primary Principal.

privileges
Privilege Attributes of the principal.

restrictions
This field enables the original owner of the PAC to impose constraints on the operations for which it is valid. There are two types of restriction:

- Mandatory: If a target to which the restriction applies cannot understand the bit string defining the restriction, access should not be granted,
- Optional: If a target application to which the restriction applies cannot understand the bit string, it is expected to ignore it.

For CSI-ECMA, it is not mandatory to generate restrictions, but mandatory restrictions cannot be ignored. If not understood, the PAC cannot be accepted.

miscellaneousAtts
Security attributes which are neither privileges attributes nor restrictions attributes. In a PAC, this may include identity attributes such as Audit Identity.

For the CSI-ECMA specification, this is the only miscellaneous attribute expected.

timePeriods
This field adds further time restrictions to the validity field of the commonContents. Either startTime or endTime can be optional. The TimePeriods control is passed if the time now is within any of the sequence periods, or if there is a period with a start before now and no endTime, or there is a period with an end after now and no startTime.

issuerIdentity
The identity of the issuing authority for the certificate.

serialNumber
The serial number of the certificate (PAC) as allocated by the issuing authority.

creationTime
The UTC time that the certificate was created, according to the authority that created it.

validity
A pair of start and end times within which the certificate is deemed to be valid.

algId
The identifier of the secret or of the public cryptographic algorithm used to seal or to sign the certificate. If there is a single identifier for both the encryption algorithm and the hash function, it appears in this field.

hashAlgId
The identifier of the hash algorithm used in the seal or in the signature.

The certificate can be uniquely identified by a combination of the issuerDomain, issuerIdentity and serialNumber.

6.7.2 Specific Certificate Contents for PACs

```
SpecificContents ::= CHOICE{
    pac                [1]   PACSpecificContents
    -- only the PAC is used here
}
PACSpecificContents ::= SEQUENCE{
    pacSyntaxVersion   [0]  INTEGER{ version1 (1)}   DEFAULT 1,
    protectionMethods  [2]  SEQUENCE OF MethodGroup   OPTIONAL,
    pacType            [4]  ENUMERATED{
                              primaryPrincipal  (1),
                              temperedSecPrincipal (2),
                              untemperedSecPrincipal(3)
                                          }    DEFAULT 3,
    privileges         [5]   SEQUENCE OF PrivilegeAttribute,
    restrictions       [6]  SEQUENCE OF Restriction     OPTIONAL,
    miscellaneousAtts  [7] SEQUENCE OF SecurityAttribute  OPTIONAL,
    timePeriods        [8]   TimePeriods                OPTIONAL
}
PrivilegeAttribute ::= SecurityAttribute
Restriction ::= SEQUENCE {
    howDefined [0] CHOICE {
                   included        [3] BIT STRING
```

```
CertificateBody ::= CHOICE{
    encryptedBody    [0]   BIT STRING,
    normalBody       [1]   SEQUENCE{
                               commonContents[0]CommonContents,
                               specificContents[1]SpecificContents
                           }
}
```

The next sections describe these three main structural components of the Generalised Certificate.

6.7.1 Common Contents fields

```
 CommonContents ::= SEQUENCE{
    comConSyntaxVersion  [0]  INTEGER { version1 (1) }DEFAULT 1,
    issuerDomain        [1]   Identifier             OPTIONAL,
    issuerIdentity      [2]   Identifier,
    serialNumber        [3]   INTEGER,
    creationTime        [4]   UTCTime                OPTIONAL,
    validity            [5]   Validity,
    algId               [6]   AlgorithmIdentifier,
    hashAlgId           [7]   AlgorithmIdentifier    OPTIONAL
}
```

Note: In the imported definition of AlgorithmIdentifier, ISO currently permits both a hash and a cryptographic algorithm to be specified. If this is done, they must appear in the algId field. The hashAlgId field is present for those cases where a separate hash algorithm specification is required.

```
Validity ::= SEQUENCE {
    notBefore     UTCTime,
    notAfter      UTCTime
} -- as in [ISO/IEC 9594-8]
 -- Note: Validity is not tagged, for compatibility with the
-- Directory Standard.
```

comConFieldsSyntaxVersion
Identifies the version of the syntax of the combination of the commonContents and the checkValue fields parts of the certificate.

issuerDomain
The security domain of the issuing authority. Not required if the form of issuerIdentity is a full distinguished name, but required if other forms of naming are in use. In CSI-ECMA, this is always a directoryName.

6.6.4 Qualifier Attributes

When a targetQualification or delegateTargetQualification method is present in the PAC, the syntax used for the method parameters is securityAttribute. Object Identifiers for qualifier attributes have the value 1.3.12.1.46.5.<qualifier attribute type>.

Currently, only one form of qualifier attribute is defined, and this identifies the target by security name. Note: this is usually the name of an identity domain as defined in CORBASEC, not an individual object. (In future, other forms of qualifier attributes may be added. For example, the attribute could identify an invocation delegation domain, rather than particular named target. Support for this would be aided by extensions to the administrative interface to the invocation delegation policy defined in CORBASEC.)

Target Names

Within a PAC protection method a target name is indicated using the OID:

```
target-name-qualifier OBJECT IDENTIFIER ::= {qualifier-attribute 1 }
```

It's syntax in the PAC is:

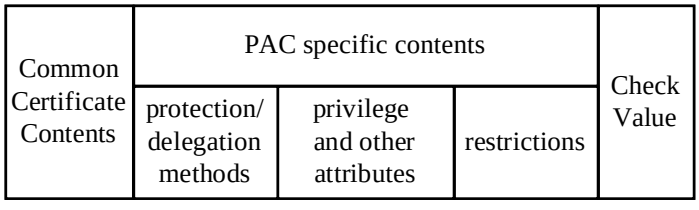
```
TargetNameValueSyntax ::= Identifier
```

6.7 PAC Format

The PAC is in the form of a generalised certificate.

A Generalised Certificate is composed of three main structural components:

- The "commonContents" fields collectively serve to provide generally required management and control over the use of the PAC.
- The "specificContents" fields are different for different types of certificate, and contain a type identifier to indicate the type. In this specification only one type is defined: the Privilege Attribute Certificate (PAC).
- The "checkValue" fields are used to guarantee the origin of the certificate. This is a signature in the CSI-ECMA specification. (though a seal would be possible as in ECMA 235)




```
GeneralisedCertificate ::= SEQUENCE{
    certificateBody  [0]   CertificateBody,
    checkValue       [1]   CheckValue}
```

- then the "family" for privilege, miscellaneous or qualifier attributes (4,3 or 5)
- then the value for that particular attribute type

All standard attributes which conformant ORBs must be able generate/transmit have this form.

In addition, conformant ORBs must be able to handle other attribute types defined here and in CORBASEC. They must also be able to handle attribute types with "OMG" object identifiers, once OMG is registered in the ISO Object Identifier hierarchy as described in Chapter 2. In this case, the Object Identifier is:

`<iso>..<omg>.<security><family definer>.<family>.<attribute type>`

where the values of the CORBA family definer, CORBA family and attribute type are as defined in CORBASEC. For standard attributes, the family definer is 0 and the family is 0 for privileges and 1 for miscellaneous attributes.

OMG Object Identifiers can also be used for privilege attributes defined by other organisations, who have registered a family definer with OMG.

6.6.3 Privilege and Miscellaneous Attribute Definitions

As described above, privilege and miscellaneous attribute types are normally identified by Object Identifiers which have a standard part, then family and attribute type parts.

The following privilege and miscellaneous attributes are defined in the CORBA Security specification and have defined attribute types. Some of these are mandatory for a CSI level 2 conformant ORB to generate - see chapter 2. The Object Identifier in the privilege attribute set for that type as follows:

Type of Attribute	oid family & type	Syntax	Meaning
access-identity	4.2	printableString	The access identity represents the principal's identity to be used for access control purposes.
primary-group	4.3	printableString	The primary group represents a unique group to which a principal belongs. A security context must not contain more than one primary group for a given principal.
group	4.4	SEQUENCE OF printableString	A group represents a characteristic common to several principals. A PAC may contain more than one group for this principal.
role	4.1	printableString	A role attribute represents one of the principal's organisational responsibilities.
audit_id	3.2	printableString	The identity of the principal as used for auditing

```
}
SecurityValue ::= CHOICE{
    directoryName      [0]   Name,
    printableName      [1]   PrintableString,
    octets             [2]   OCTET STRING,
    intVal             [3]   INTEGER,
    bits               [4]   BIT STRING,
    any               [5]   ANY -- defined by attributeType }
```

Note: only one set member is permitted in AttributeValue. Multivalue attributes are effected in the securityValue field, where the "SEQUENCE OF" construct can be used. (Including "SET OF" in the syntax enables security attributes to be stored as normal in a Directory whenever the choice made within Identifier is OBJECT IDENTIFIER.)

Note: a directory name is translated into a string format as defined in 6.4. above. So the octet string attribute value returned at the IDL interface is a representation of this string, not the more complex ASN.1 definition of this.

attributeType
Defines the type of the attribute. Attributes of the same type have the same semantics when used in Access Decision Functions, though they may have different defining authorities.

definingAuthority
The authority responsible for the definition of the semantics of the value of the security attribute. This optional field of the attributeValue can be used to resolve potential value clashes. It is defined as an Identifier which has a choice of syntax. For CSI-ECMA, it is always a directoryName.

securityValue
The value of the security attribute. Its syntax is can be either one of the basic syntaxes for attributes or a more complex one determined by the attribute type.

6.6.2 Attribute Types

An attribute type in this standard is formally defined as an Identifier which provides a choice of syntax. However, all standard attribute types are defined as OBJECT IDENTIFIERs. Three types of attributes are defined:

- privilege attributes e.g. access_id, group, role
- miscellaneous attributes, mainly the audit_id
- qualifier attributes used within the PV/CV delegation scheme to say where credentials can be used/delegated

For standard attributes, the OBJECT IDENTIFIER includes:

- a standard part with the value 1.3.12.1.46

```
            -- Dialogue Key. Contains only the sealValue field
}
CDTContents ::= SEQUENCE {
    tokenType     [0]   OCTET STRING VALUE X'0301',
    SAId          [1]   OCTET STRING,
    utcTime       [2]   UTCTime OPTIONAL,
    usec          [3]   INTEGER OPTIONAL,
    seq-number    [4]   INTEGER OPTIONAL,
}
```

cdtContents
This contains only administrative fields, identifying the token type, the context and providing exchange integrity.

seq-number
When present, this field contains a value one greater than that of the seq-number field of the last token issued from this issuer.

The other administrative fields are as described above.

trtSeal
See above for a general description of the use of this construct.

6.6 Security Attributes

6.6.1 Data Structures

The security attribute is a basic construct for privilege and other attributes in PACs.

```
SecurityAttribute ::= SEQUENCE{
    attributeType      Identifier,
    attributeValue     SET OF SEQUENCE {
                       definingAuthority [0] Identifier OPTIONAL,
                       securityValue     [1] SecurityValue}
}
Identifier ::= CHOICE{
    objectId           [0]   OBJECT IDENTIFIER,
    directoryName      [1]   Name,
            -- imported from the Directory Standard
    printableName      [2]   PrintableString,
    octets             [3]   OCTET STRING,
    intVal             [4]   INTEGER,
    bits               [5]   BIT STRING,
    pairedName         [6]   SEQUENCE{
                               printableName [0] PrintableString,
                               uniqueName    [1] OCTET STRING  }
```

MICToken

A MICToken is a per-message token, separate from the user data being protected, which can be used to verify the integrity of that data as received. The token is passed in the message_protection_token in SECIOP messages, and the protected data follows as a GIOP message or message fragment. The syntax of the token is:

```
MICToken  ::=   PMToken
```

The overall structure and field contents of the token are described above. Fields specific to the MICToken are:

userData
Not present for MIC Tokens.

pmtSeal
The Checksum is calculated over the DER encoding of the pmtContents field with the user data temporarily placed in the userData field. The userData field is not transmitted.

WrapToken

A WrapToken encapsulates the input user data (optionally encrypted) along with associated integrity check values. It consists of an integrity header followed by a body portion that contains either the plaintext or encrypted data. The syntax of the token is:

```
WrapToken  ::=   PMToken
```

The overall structure and field contents of the token are described above. Fields specific to the WrapToken are:

userData
Present either in plain text form, or encrypted. If the data is encrypted, it is performed using the Confidentiality Dialogue Key, and as in [KRBV5], an 8-byte random confounder is first prepended to the data to compensate for the fact that an IV of zero is used for encryption.

wtSeal
The Checksum is calculated over the pmtContents field, including the userData. However if the userData field is to be encrypted, the seal value is computed prior to the encryption.

6.5.5 ContextDeleteToken

The ContextDeleteToken is issued by either the context initiator or the target to indicate to the other party that the context is to be deleted.

```
ContextDeleteToken ::=  SEQUENCE {
    cdtContents  [0]   CDTContents,
    cdtSeal      [1]   Seal
            -- seal over cdtContents, encrypted under the Integrity
```

```
PMTContents ::= SEQUENCE {
    tokenId             [0]   INTEGER, -- shall contain X'0101'
    SAId                [1]   OCTET STRING,
    seq-number          [2]   INTEGER                 OPTIONAL,
    userData            [3]   CHOICE {
                                plaintext    BIT STRING,
                                ciphertext   OCTET STRING
                                }                     OPTIONAL,
    directionIndicator [4]    BOOLEAN                 OPTIONAL
}
```

pmtContents

tokenId
SAId
See above for a description of these fields

seq-number
This field must be present if replay detection or message sequencing have been specified as being required at Security Association initiation time. The field contains a message sequence number whose value is incremented by one for each message in a given direction, as specified by directionIndicator. The first message sent by the initiator following the InitialContextToken shall have the message sequence number specified in that token, or if this is missing, the value 0. The first message returned by the target shall have the message sequence number specified in the TargetReplyToken if present, or failing this, the value 0.

The receiver of the token will verify the sequence number field by comparing the sequence number with the expected sequence number and the direction indicator with the expected direction indicator. If the sequence number in the token is higher than the expected number, then the expected sequence number is adjusted and GSS_S_GAP_TOKEN is returned. If the token sequence number is lower than the expected number, then the expected sequence number is not adjusted and GSS_S_DUPLICATE_TOKEN or GSS_S_OLD_TOKEN is returned, whichever is appropriate. If the direction indicator is wrong, then the expected sequence number is not adjusted and GSS_S_UNSEQ_TOKEN is returned

userData
See specific token type narratives below.

directionIndicator
FALSE indicates that the sender is the context initiator, TRUE that the sender is the target.

pmtSeal

See specific token type narratives below.

6.5.3 ErrorToken

An error token may be returned as follows:

```
ErrorToken ::=   {
    tokenType     [0]   OCTET STRING VALUE X'0400',
    etContents    [1]   ErrorArgument,
}
```

etContents
Contains the reason for the creation of the error token. The different reasons are given as minor status return values.

```
ErrorArgument ::= ENUMERATED {
    gss_ses_s_sg_server_sec_assoc_open              (1),
    gss_ses_s_sg_incomp_cert_syntax                 (2),
    gss_ses_s_sg_bad_cert_attributes                (3),
    gss_ses_s_sg_inval_time_for_attrib              (4),
    gss_ses_s_sg_pac_restrictions_prob              (5),
    gss_ses_s_sg_issuer_problem                     (6),
    gss_ses_s_sg_cert_time_too_early                (7),
    gss_ses_s_sg_cert_time_expired                  (8),
    gss_ses_s_sg_invalid_cert_prot                  (9),
    gss_ses_s_sg_revoked_cert                       (10),
    gss_ses_s_sg_key_constr_not_supp                (11),
    gss_ses_s_sg_init_kd_server_ unknown            (12),
    gss_ses_s_sg_init_unknown                       (13),
    gss_ses_s_sg_alg_problem_in_dialogue_key_block  (14),
    gss_ses_s_sg_no_basic_key_for_dialogue_key_block(15),
    gss_ses_s_sg_key_distrib_prob                   (16),
    gss_ses_s_sg_invalid_user_cert_in_key_block     (17),
    gss_ses_s_sg_unspecified                        (18),
    gss_ses_s_g_unavail_qop                         (19),
    gss_ses_s_sg_invalid_token_format               (20)
}
```

6.5.4 Per Message Tokens

The syntax of the message_protection_token in SECIOP messages has the same general structure for both MIC and Wrap tokens:

```
PMToken ::=  SEQUENCE{
    pmtContents    [0]   PMTContents,
    pmtSeal        [1]   Seal
        -- seal over the pmtContents being protected
}
```

targetIdentity
The identity of the intended target of the Security Association. Used by the targetAEF to validate the PAC. Can also be used by the targetAEF to help protect the delivery of dialogue keys.

flags
flags required by the Target AEF for its validation process. Only contains a delegation flag, the value of which is the same as the value of delegation flag in contextFlag field of ictContents. When the flag is set, all ECVs sent in pacAndCVs are made available to the target. Other bits are reserved for future use.

6.5.2 *TargetResultToken*

This token is returned by the target if the mutual-req flag is set in the Initial Context Token. It serves to authenticate the target to the initiator, since only the genuine target could derive the integrity dialogue key needed to seal the TargetResultToken.

```
TargetResultToken ::=  SEQUENCE{
    trtContents    [0] TRTContents,
    trtSeal        [1] Seal
}
TRTContents ::= SEQUENCE {
    tokenId        [0]   INTEGER,    -- shall contain X'0200'
    SAId           [1]   OCTET STRING,
    utcTime        [5]   UTCTime     OPTIONAL,
    usec           [6]   INTEGER     OPTIONAL,
    seq-number     [7]   INTEGER     OPTIONAL,
}
```

Note: There is no field for returning certification data here. This is because any such data that may be required is assumed to be returned at the conclusion of mechanism negotiation.

trtContents
This contains only administrative fields, identifying the token type, the context and providing exchange integrity.

seq-number
When present, specifies the target's initial sequence number, otherwise, the default value of 0 is to be used as an initial sequence number.

The other administrative fields are as described in above.

trtSeal
Seal of trtContents computed with the integrity dialogue key. Only the sealValue field of the Seal data structure is present. The cryptographic algorithms that apply are specified by integDKUseInfo in the dialogueKeyBlock field of the initial context token.

usec
Micro second part of the initiator's time stamp. This field along with utcTime are used together to specify a reasonably accurate time stamp

seq-number
When present, specifies the initiator's initial sequence number. Otherwise, the default value of 0 is to be used as an initial sequence number.

initiatorAddress
Initiator's network address part of the channel bindings. This field is only present when channel bindings are transmitted by the caller to the mechanism implementation. Conformant ORBs do not need to generate this field).

targetAddress
Target's network address part of the channel bindings. This field is only present when channel bindings are transmitted by the caller to the implementation.

TargetAEF Part

```
TargetAEFPart ::= SEQUENCE {
    pacAndCVs          [0]   SEQUENCE OF CertandECV OPTIONAL,
    targetKeyBlock     [1]   TargetKeyBlock,
    dialogueKeyBlock   [2]   DialogueKeyBlock,
    targetIdentity     [3]   Identifier,
    flags              [4]   BIT STRING   {
                                  delegation         (0)
                                  }
}
```

pacAndCVs
The initiator ACI to be used for this Security Association. This field is not present when the association does not require any ACI. This field contains the PAC together with associated PAC protection information. When only simple delegation is supported, exactly one of these should be present.

If composite delegation options are supported, this field will contain more than one PAC. For example, for the initiator plus immediate invoker case, the initiator’s PAC would be present (with CVs) and also the immediate invoker’s (with a PPID).

targetKeyBlock
The targetKeyBlock carrying the basic key to be used for the Security Association being established.

dialogueKeyBlock
A dialogue key block used by the targetAEF along with the basic key to establish an integrity dialogue key and a confidentiality dialogue key for per-message protection over the Security Association being established.

tokenId
Identifies the initial-context token. Its value is 01 00 (hex)

SAId
A random number for identifying the Security Association being formed; it is one which (with high probability) has not been used previously. This random number is generated by the initiator and processed by the target as follows:

- If no targetResultToken is expected, the SAId value is taken to be the identifier of the Security Association being established (if this is unacceptable to the target, then an error token with etContents value of gss_ses_s_sg_sa_already_established must be generated).
- If a targetResultToken is expected, the target generates its random number and concatenates it to the end on the initiator's random number. The concatenated value is then taken to be the identifier of the Security Association being established.

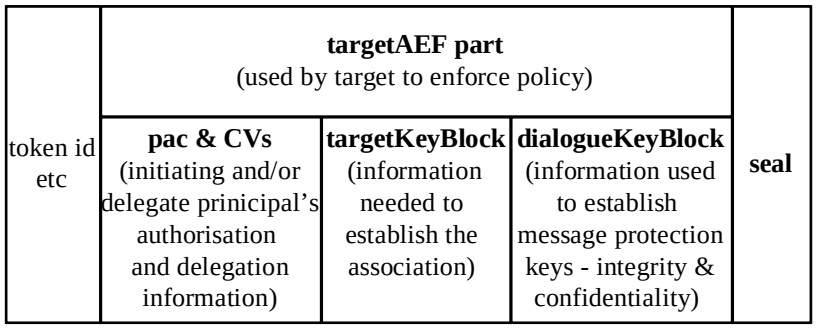
targetAEFPart
Part of the initial-context token to be passed to the target access enforcement function. This is defined below and includes PAC, basic and dialogue key packages

targetAEFPartSeal
Seal of the targetAEFPart computed with the basic key. Only the sealValue field of the Seal data structure is present. The cryptographic algorithms that apply are specified by algorithm profile in the mechanism option

contextFlags
Combination of flags that indicates context-level functions requested by the initiator.

delegation	when set to 0, indicates that the initiator explicitly forbids delegation of the PAC in the targetAEFPart.
mutual-auth	indicates that mutual authentication is requested.
replay-detect	indicates that replay detection features are requested to be applied to messages transferred on the established Security Association.
sequence	indicates that sequencing features are requested to be enforced to messages transferred on the established Security Association.
conf-avail	indicates that a confidentiality service is available on the initiator side for the established Security Association.
integ-avail	indicates that an integrity service is available on the initiator side for the established Security Association.

utcTime
The initiator's UTC time.




```
 InitialContextToken ::=  SEQUENCE{
    ictContents     [0]   ICTContents,
    ictSeal         [1]   Seal
}
```

ictContents
Body of the initial context token

ictSeal
Seal of ictContents computed with the integrity dialogue key. Only the sealValue field of the Seal data structure is present. The cryptographic algorithms that apply are specified by integDKUseInfo in the dialogueKeyBlock field of the initial context token.

```
ICTContents ::= SEQUENCE {
    tokenId            [0]   INTEGER, -- shall contain X'0100'
    SAId               [1]   OCTET STRING,
    targetAEFPart      [2]   TargetAEFPart,
    targetAEFPartSeal  [3]   Seal,
    contextFlags       [4]   BIT STRING {
                                 delegation         (0),
                                 mutual-auth        (1),
                                 replay-detect      (2),
                                 sequence           (3),
                                 conf-avail         (4),
                                 integ-avail        (5)
                             }
    utcTime            [5]   UTCTime       OPTIONAL,
    usec               [6]   INTEGER       OPTIONAL,
    seq-number         [7]   INTEGER       OPTIONAL,
    initiatorAddress   [8]   HostAddress   OPTIONAL,
    targetAddress      [9]   HostAddress   OPTIONAL
}
```

6.5 SECIOP tokens when using CSI-ECMA

All SECIOP security tokens conform to the basic token format defined in 3.4.1. The object identifier for the MechType is of the form:

```
{ generic_CSI_ECMA_mech  (y) (z) }
```

where the value for generic_CSI_ECMA_mech is 1.3.12.0.235.4 and the values of y and z, if present, represent the architectural option number and cryptographic profile numbers as described above. Both y and z can be defaulted.

The innerContextToken of the SECIOP message may be any of the tokens defined in Chapter 3 above. Therefore, for context establishment, tokens are:

InitialContextToken	sent by the initiator to a target, to start the process of establishing a Security Association.
TargetResultToken	sent to the initiator by the target, if needed, following receipt of an Initial Context Token.
ErrorToken	sent by the target on detection of an error during Security Association establishment.

The per-message tokens are:

MICToken	sent either by the initiator or the target to verify the integrity of the user data sent separately.
WrapToken	sent either by the initiator or the target. Encapsulates the input user data (optionally encrypted) along with integrity check values.

A ContextDeleteToken may also be used by either by the initiator, or the target to release a Security Association.

This definition uses ASN.1 types from other standards, for example, the ISO definition of a Certificate. These types are detailed in Annex E of ECMA-235.

6.5.1 Initial Context Token

The initial context token contains:

- general information such as the token id, contextFlags (delegation, replay-detect etc), utcTime, seq-number etc
- a targetAEF part to be passed to the target access enforcement function. This includes the PAC and associated CVs, target key block and dialogue key package
- a seal

6.4 Security Names

This protocol uses two forms of security names.

- Directory names (DNs) are used where public key technology is used, as this is the form of name used in X509 certificates
- Kerberos names are used where secret key technology is used, as this is the form of name used by Kerberos

Kerberos Naming

An entity that uses the normal Kerberos V5 authentication is given a printable Kerberos principal name of the form:

```
<principal_name>@realm_name>
```

Notes:

1 Components of a name can be separated by "/".

2 The separator @ signifies that the remainder of the string following the @ is to be interpreted as a realm identifier. If no @ is encountered, the name is interpreted in the context of the local realm. Once an @ is encountered, a non-null realm name, with no embedded "/" separators must follow.The "/" character is used to quote the immediately-following character.

Directory Naming

Where public key technology supported by Directory Certificates is used, entities are given DNs. Such names are normally transmitted as directoryNames. At interfaces, they are strings built from components separated by a semicolon. The standardised keywords supported are:

```
CN  (common-name)
S   (surname)
OU  (organisation unit)
O   (organisation)
C   (country)
```

So an example of a supported DN is:

```
CN=Martin;OU=Sesame;O=Bull;C=fr
```

Note that there is no general rule for mapping the Directory name of an entity onto its Kerberos principal name, so an explicit mapping is provided in a principal's Directory Certificate, using the extensions field of the extended Directory Certificate syntax (version 3) to carry the principal's Kerberos name.

The syntax of the login name is imported from the Kerberos V5 GSS-API mechanism. This the form of name is referred to using the symbolic name: GSS_KRB5_NT_PRINCIPAL. Syntax details are given in [KRB5GSS].

A PAC will be accepted by the target (subject to other controls in the PV's method group) if the client proves knowledge of the CV by passing it (encrypted) as part of the initial context token. A method group contains at most one PV value.

In the simplest case, the method group contains just the PV and the target can delegate the PAC if it receives the CV.

The PV/CV method can be used for more selective targeting of the PAC also. A method group can include qualifier attributes which specify where the PAC can be used. Qualifier attributes can specify which principals can receive the PAC as a target and which can act as both delegate and target. These principals can be specified by their identities (though the protocol is extensible for other options such as a group/domain to which they belong).

Note that as for the simpler case, delegation can be prevented by setting the delegation mode to NoDelegation (see CORBASEC). This will cause the client to send the PAC without the CV.

[The protocol allows more than one method group in the PAC, each with its own PV/CV. This can be used by a client or intermediate object in a chain to further restrict who can use the PAC, by failing to send some of the CVs. However, the current CORBASEC specification does not include any IDL for restricting delegation in this way, so it is not be possible to exploit this capability.]

6.2.10 Restrictions

Other restrictions may be included in the PAC. An ORB conforming to this specification does not need to generate these restrictions, but will reject PACs with mandatory restrictions which it does not understand or cannot process.

6.3 Mechanism Identifiers and IOR Encoding

Mechanism identifiers for the CSI-ECMA protocol have up to three parts as follows:

- the *protocol identifier*. This is CSI-ECMA.
- the *architectural option*. This identifies the architectural option, i.e. the key distribution method used when establishing security associations.
  If absent, the default option is used
- the *cryptographic profile*. This identifies the cryptographic profile as defined above.
  If absent, a default is used.

In the IOR, the mechanism name in the struct of the TAG_x_SEC_MECH is:

`CSI-ECMA_<architectural option>`

where the architectural options supported are Secret, Hybrid and Public, so mechanism names are CSI_ECMA_Secret, CSI_ECMA_Hybrid and CSI_ECMA_Public.

These values could also be negotiated using a generic mechanism negotiation scheme such as that in [SNEGO] in future, but are in the IOR for the current CSI specification.

6.2.7 PAC Protection and Delegation - Outline

The ECMA protocol provides a number of ways of protecting a principal's credentials as held in a PAC. In CSI-ECMA, a digital signature. is used, as this allows a target system to check what Security Authority authorised use of these privileges, without relying on the transitive trust needed for sealed PACs crossing domain boundaries. Encrypted PACs are not included in this profile.

There may also be controls on where the PAC may be delegated and used.

Protection method fields in the PAC specify where this PAC can be used and whether it can be used by the specified targets only (for example, allowing use of the privileges for access control) or whether that target can also delegate it.

Protection method fields are grouped together into method groups. The protection method check is passed if all the method fields in any one of the method groups is passed.

6.2.8 PPID Method

This method protects the PAC from being stolen, by restricting the initiators who can use the PAC.

When no other method group is present, it permits the PAC to be used only by the client entity to which it was originally issued i.e. it prevents delegation. However, a PAC with a PPID will be delegatable if delegation is permitted by a PV/CV method - see below.

A PPID identifying the initiating principal is put in the PAC by the Privilege Attribute (or other security) Service, according to policy or client request. The same/related information is also supplied as part of the targetKeyBlock so the target can check that the entity which sent this token is the same entity which is entitled to use the PAC.

The PPID is a security attribute whose value in the CSI-ECMA protocol can take one of two forms, depending on the key distribution scheme used by the initiator.

- When the initiator has a secret key, the PPID is a random bit string which is also sent in the authorization field of the Kerberos ticket. This ticket is sent as part of the targetKeyBlock and can be checked to come from this client
- For the public key scheme, the PPID contains the certificate serial number and CA name for the initiator's X.509 public key certificate. The targetKeyBlock sent to the target is signed using this initiator's private key.

6.2.9 PV/CV Delegation Method

This method prevents the PAC from being stolen and at the same time controls whether (and where) it can be delegated. The method field in the PAC contains a protection value (PV) which is a one way function of a Control Value (CV).

The algorithms can now be further categorised into broader classes as follows:

Class 1:	symmetric for security of mechanism:	uses 3, 5, 7
Class 2:	all OWFs:	uses 2, 4, 6, 8, 11
Class 3:	internal mechanism asymmetric, encrypting:	use 9
Class 4:	internal mechanism asymmetric, non encrypting:	use 2
Class 5:	CA's asymmetric non-encrypting:	use 6
Class 6:	data confidentiality, symmetric:	use 12

Use 10 is a fixed value, and does not contribute to mechanism use options.

Based on these classes, the following cryptographic algorithm usage profiles are defined. Other profiles are possible and can be defined as required. Note that symmetric algorithm key sizes are included in this profiling, thus DES/64 indicates DES with a 64 bit key.

	Profile 1 Full	**Profile 2 no data confidentiality**	**Profile 3 low grade confidentiality**	**Profile 5 defaulted**
Class 1	DES/64	DES/64	RC4/128	separately agreed default
Class 2	MD5	MD5	MD5	separately agreed default
Class 3	RSA	RSA	RSA	separately agreed default
Classes 4 and 5	RSA	RSA	RSA	separately agreed default
Class 6	DES/64	None	RC4/40	separately agreed default

Where:

- Profile 1 provides full security, using standard cryptographic algorithms with common accepted key sizes.
- Profile 2 is the same but without supporting any confidentiality of user data.
- Profile 3 provides low grade confidentiality. In some countries, products using this are exportable without restriction; in others, they are more easily exportable/importable.
- Profile 5 uses algorithms identified by a separately specified default. It is intended for use by organisations who wish to use their own proprietary or government algorithms by separate agreement or negotiation.

Full public key scheme

In this scheme, both client and target posses private/public keys. Neither use a KDS. The scheme name for this is: *asymmetric*. The architectural option number is 6.

To establish the client-target association, the client constructs a targetKeyBlock containing a basic key encrypted under the target's public key. The target key block is signed with the client's private key. On receipt of the targetKeyBlock, the target directly establishes a basic key from it.

6.2.6 Cryptographic Algorithms and Profiles

Cryptographic and hashing algorithms are used for various purposes. This section categorises the algorithms according to usage so that client and targets can more easily determine if they have the cryptographic support required to allow interoperation. The categorisation is then refined into cryptographic profiles that can be incorporated into specific mechanism identifiers.

The mechanism identifiers with cryptographic profiles can then be carried in the IOR.

The table below summarises the different uses to which algorithms are put.

Use Reference	Description of Use	Type of Algorithm
2	PAC protection using signature	OWF + asymmetric signature
3	basic key usage	confidentiality and integrity
4	integrity dialogue key derivation	OWF
5	integrity dialogue key usage	symmetric integrity
6	CA public keys	OWF + asymmetric signature
7	encryption using shared long term symmetric key	symmetric confidentiality
8	name hash to prevent ciphertext stealing	OWF
9	asymmetric basic key distribution	asymmetric encryption
10	key establishment within SPKM_REQ	(fixed value)
11	confidentiality dialogue key derivation	OWF
12	confidentiality dialogue key use	symmetric confidentiality

be configured. The information required to derive the dialogue keys is transmitted in the Dialogue key package. Typically, dialogue keys are constructed from the basic key using a one way algorithm.

6.2.5 Key Distribution Schemes

The CSI-ECMA protocol allows a choice of key distribution methods for establishing a client-target security association including the basic key. The content of the targetKeyBlock depends on the scheme used.

The key distribution schemes depend on the existence of long term cryptographic keys. Both secret (symmetric) and public (asymmetric) key technology can be used. When secret keys are used, a key is shared between the target and its Key Distribution Service (KDS). When public keys are used, the private key is kept by the principal and the public key held in a certificate, in a directory or elsewhere.

Initiators may also possess symmetric or asymmetric keys established as the result of an earlier authentication.

This CSI-ECMA specification defines three key distribution schemes. These are described below and are identified by a name and an architectural option number. Other schemes are possible as extensions to this as described in ECMA-235.

Basic symmetric key distribution scheme

In this scheme, the client and target each share different secret keys with the same Key Distribution Server. The scheme name for this is: *symmIntradomain*. The architectural option number is 2.

To establish the association, between the client and target, the client obtains a targetKeyBlock from its KDS containing a basic key encrypted under the target's long term key. On receipt of the targetKeyBlock, the target can extract the basic key from it.

In this case, the targetKeyBlock is a Kerberos ticket.

Symmetric key distribution with asymmetric KDSs

In this scheme, the initiator shares a secret key with its KDS and the target shares a secret key with its KDS (which is different). In addition, each KDS possesses a private/public key pair. The scheme name for this is: *hybridInterdomain*.The architectural option number is 3.

To establish the client-target association, the client gets a targetKeyBlock from its KDS containing the basic key encrypted under a temporary key and the temporary key encrypted under the target's KDS's public key. The targetKeyBlock is also signed using the initiator KDS's private key.

On receipt of the targetKeyBlock, the target transmits it to its KDS and gets back the basic key encrypted under the long term secret key it shares with its KDS.

In line with the CORBA Security specification, each privilege attribute has a defining authority which identifies the authority responsible for defining the semantics of the value of the security attribute. This can be included for each privilege attribute in the PAC and in this case, there could be a different defining authority for each privilege.

It is often the case that most attributes in the PAC come under the same defining authority and this is the authority which issued the PAC. If the PAC as transmitted does not have defining authorities for some attributes, then the issuing authority of the PAC is considered to be the defining authority.

Miscellaneous attributes

This specification allows other types of security attributes to be carried in the PAC under the general heading of miscellaneous attributes. In CSI-ECMA, the only type of miscellaneous attribute supported is the audit identity.

6.2.3 Target Access Enforcement Function

The security processing functionality at the target is split between the target application and the target access enforcement function (targetAEF). ISO (ISO/IEC 10181-3) defines an access enforcement function collocated with the target application which controls access to a target application. This has a number of advantages including:

- security critical code is isolated which makes security evaluation simpler
- long term keys can be shared between applications/objects. This can simplify administration (as there are less keys) and allow re-use of keying information when accessing another application/object sharing this targetAEF.

The targetAEF is responsible for setting up the security association, including validating the PAC, and releasing the keys for message protection.

6.2.4 Basic and Dialogue Keys

The exchanges between client and target are secured using a two level key scheme in which a distinction is made between basic and dialogue keys.

A basic key is a temporary key established between a client and the target (actually, the targetAEF). The basic key is used for integrity protection of the PAC and associated information, its own key establishment information and the information used to establish the dialogue keys. The basic key is established by the client sending information to the target in the targetKeyBlock. This can take different forms depending on the key distribution method used - see below.

A dialogue key is a temporary key established between the client and target and used to protect the requests and responses. Separate dialogue keys can be established for integrity and confidentiality protection, enabling different strengths of mechanism to

6.2 Concepts

6.2.1 Separation of Concerns

As outlined above, the initial context token transmitted in the SECIOP EstablishContext message on setting up a security association contains a number of parts with limited links between them. This is so that the different parts can be varied reasonably independently of each other. The three main parts are:

- authorisation information - the Privilege Attribute Certificate (PAC) which contains the privileges used for access control and other attributes such as the audit id. Associated with this are delegation and other controls.
  This is therefore concerned with the access control and delegation policies, but is mainly independent of the key establishment and message protection mechanisms. So this can be updated to affect these policies independently of mechanisms. (The size of the PAC may be significant, so it is not confidentiality protected, as this may cause regulatory problems - see 1.4.2.)

  Privilege and other attributes in PACs are described in 6.2.2 below

- target key block - used to provide the information needed to establish the security association between client and target. Secret key or public key technology (or some hybrid of these) may be used. However, the result is always a "basic" key from which dialogue keys to protect application messages can be derived.
  This is therefore concerned with the mechanism for establishing trust and distributing keys. This can be varied independently of the authorisation policies and the message protection methods

  Key establishment methods are described in 6.2.5 below.

- dialogue key packages which control how dialogue keys to protect messages are derived from the basic key.
  Note that this is largely independent of the key distribution method. i.e. public key technology may be used to establish secret keys for dialogue protection.

6.2.2 Security Attributes

Privilege Attributes

The CSI-ECMA protocol allows a range of privilege attributes in a Privilege Attribute Certificate (PAC) transmitted between the client and target object. These privileges can then be used for access control.

Privilege attributes which can be carried in the PAC at level 2 are defined in Chapter 2 and include all those defined in the CORBA security specification.

A vendor or user enterprise may also define its own privilege attributes (if the particular implementation allows this) and use them for access control.

CSI-ECMA Protocol 6

6.1 Introduction

This chapter defines the CSI-ECMA protocol. It is based on the ECMA GSS-API mechanism as defined in ECMA-235, though is a significant subset of that. It supports all CSI functionality levels (0, 1 and 2). It provides three options for key distribution:

- a secret key option using Kerberos data structures
- a hybrid option where secret keys are used within an administrative domain, but public keys are used between domains
- a public key option which uses public key technology for key distribution both within and between domains

This chapter includes the full definition of the CSI-ECMA protocol so it can be read without reference to ECMA 235 - the standard on which it is based. (ECMA-235 contains a lot of material not relevant to this standard, including further key distribution options and also APIs not needed in a CORBA environment, where the IDL interfaces specified in CORBASEC are used.) The CSI-ECMA protocol is very similar to the SESAME profile as described in {SESAMEMECH].

The CSI-ECMA protocol supports the level 2 facilities in the CORBA Security specification. It is designed to be extensible as new facilities, for example, new delegation options, are agreed in future, and also further key distribution options. It is also designed to respond to the requirements of international deployment such as minimal confidentiality (only keying information needs to be encrypted), use of anonymous audit (a separate audit_id can be transmitted), choice of cryptography for message protection (including strong integrity, weak confidentiality).

The structure of the initial context token is key to providing this flexibility. It is separated into 3 parts:

- authorisation information
- information concerned with establishing the security association using one of the supported key distribution options
- information concerned with generating the dialogue keys for message protection

5

The GSS_C_DELEG_FLAG is set when either the client has called *set_security_features* specifying SimpleDelegation or when an administrator has called *set_delegation_mode* with a value of SimpleDelegation on a domain to which the target object belongs. The optional “Deleg” field, if present, includes a forwardable Ticket Granting Ticket (TGT) representing the delegated credentials of the client sending the EstablishContext message.

The GSS_C_MUTUAL_FLAG is set when either the client has called *set_association_options* specifying a value of EstablishTrustInTarget or an administrator has called *set_association_options* with a value of EstablishTrustInTarget on the domain to which the target belongs.

The GSS_C_REPLAY_FLAG and GSS_C_SEQUENCE_FLAG are generally clear as they can cause incorrect replay and misordering detection in a multi-threaded environment - see section 3.3.2. [Note also, that the current GSS Kerberos implementation available without cost from MIT does not support replay detection.]

The Final Context Token

The *final_context_token* carried within a CompleteEstablishContext SECIOP message is encoded according to the formats defined in [GSSKRB5] Section 1.1.2.

The Continuation Context Token

Kerberos V5 does not use the ContinueEstablishContext message and therefore does not define the *continuation_context_token* format. If the Kerberos V5 mechanism is amended in the future to support mechanism negotiation, support of the ContinueEstablishContext message would be necessary and thus definition of the *continuation_context_token* would be required.

The Message Protection Token

The *message_protection_token* carried within a SECIOP MessageInContext message is encoded according to the formats defined in [GSSKRB5] section 1.2.

Mandatory and Optional Cryptographic Profiles

ORB implementations claiming conformance to the GSS Kerberos protocol must implement at least the MD5 profile. Conformant ORBs may, but are not required to implement the remaining cryptographic profiles defined in this specification.

5.3 IOR Encoding

The security tags in the IOR are encoded as described in Chapter 3. Both security name and association options tags may appear in the IOR and be shared between mechanisms.

The component data member associated with the KerberosV5 mechanism tag is a struct defined as follows:

```
struct KerberosV5 {
AssociationOptions                target_supports;
AssociationOptions                target_requires;
sequence<CryptographicProfile>    crypto_profiles;
sequence<octet>                   security_name;
}
```

`security_name` shall contain a valid Kerberos Principal Name of type GSS_KRBV5_NT_PRINCIPAL_NAME, which is defined in [GSSKRB5].

The association options are as defined in 3.5.2 above.

5.4 SECIOP Tokens

When the GSS-Kerberos protocol is chosen as the security mechanism for invoking an object, the SECIOP protocol carries the information described in this section.

All Kerberos tokens are encoded according to the general format described in 3.6. The OBJECT IDENTIFIER for Kerberos V5 is 1.3.5.1.2 until [GSSKRB5] is advanced to a Proposed Standard RFC when it will be changed to 1.2.840.113554.1.2.2.

Each individual token is distinguished by the data carried in the ANY field of this general framework.

The Initial Context Token

The *initial_context_token* carried within an EstablishContext SECIOP message is encoded according to the general framework and conforms to the unencrypted authenticator message as described in [GSSKRB5] Section 1.1.1.

Note that channel bindings are required to be ZERO (GSS_C_NO_BINDINGS) in this specification - see section 3.4.3 above..

GSS Kerberos Protocol 5

5.1 Introduction

This chapter specifies the GSS Kerberos protocol. It is based on the GSS Kerberos specification [GSSKRB5] which itself is based on Kerberos V5 as defined in [KERBV5]. This specification refers to, rather than repeats, information in [GSSKRB5] and [KERBV5].

This chapter defines the required information for encoding the mechanism specific information in the IOR and the token formats used by the SECIOP protocol.

5.2 Cryptographic Profiles

The following cryptographic profiles are supported with this mechanism:

DES_CBC_DES_MAC
Specifies use of the Kerberos V5 mechanism with DES MAC message digest for integrity and DES in CBC mode for confidentiality.

DES_CBC_MD5
Specifies use of the Kerberos V5 mechanism with MD5 message digest for integrity and DES in CBC mode for confidentiality.

DES_MAC
Specifies use of the Kerberos V5 mechanism with DES MAC message digest for integrity.

MD5
Specifies use of the Kerberos V5 mechanism with a DES encrypted MD5 message digest for integrity.

Values for these cryptographic profiles are assigned in A.2.

4

detection during the context, if this has been requested by the application). SPKM_1 OBJECT IDENTIFIER is 1.3.6.1.5.5.1.1 and SPKM_2 OBJECT IDENTIFIER is 1.3.6.1.5.5.1.2.

The Initial Context Token

The *initial_context_token* carried within an establishContext SECIOP message is encoded according to the general framework and confirms to the SPKM-REQ token as described in [SPKMMECH] Section 3.1.1.

In the initial_context_token, channel bindings are required to be ZERO (GSS_C_NO_BINDINGS).

The GSS_C_DELEG_FLAG is required to be FALSE (no delegation is supported).

The GSS_C_MUTUAL_FLAG is TRUE if it requires both parties to authenticate itself and FALSE (the default) if only one party is required to authenticate itself.

The Final Context Token

The *final_context_token* carried within a CompleteEstablishContext SECIOP message is encoded according to the SPKM-REP-TI token as defined in [SPKMMECH] Section 3.1.2 or the SPKM-ERROR token as defined in [SPKMMECH] Section 3.1.4.

The Continuation Context Token

The *continuation_context_token* carried within a ContinueEstablishContext SECIOP message is encoded according to the SPKM-REP-TI token or the SPKM-REP-IT token as defined in [SPKMMECH] Section 3.1.3 or the SPKM-ERROR token.

The Message Protection Token

The *message_protection_token* carried within a SECIOP MessageInContext message is encoded according to the SPKM-MIC token (for integrity) or SPKM-WRAP token (for confidentiality) as defined in [SPKMMECH] Section 3.2.

The Context Delete Token

The *context_delete_token* carried within a SECIOP DiscardContext message is encoded according to the SPKM-DEL token as defined in [SPKMMECH] Section 3.2.3. This assumes DiscardContext messages can include a *discard_context_token* - see B.3.2.

MD5_DES_CBC_SOURCE
Specifies use of the SPKM mechanism to provide data integrity by encrypting, using DES in CBC mode, the MD5 hash of that data. The default key establishment algorithm is used plus source authentication information is also encrypted with the target's public key.

DES_CBC_SOURCE
Specifies use of SPKM mechanism to provide data confidentiality by using DES in CBC mode. The default key establishment algorithm is used plus source authentication information is also encrypted with the target's public key.

Values for these cryptographic profiles are assigned in A.2.

4.3 IOR Encoding

The security tags in the IOR are encoded as described in Chapter 3.

The component data member associated with the SPKM_1 and SPKM_2 mechanism tags is a struct defined as follows:

```
struct <mechanism_name> {
   AssociationOptions                target_supports;
   AssociationOptions                target_requires;
   sequence <cryptographicProfile>   crypto_profiles;
   sequence<octet>                   security_name; }
```

`mechanism_name` can be either SPKM_1 or SPKM_2 and `security_name` must contain a valid X.500 distinguished name represented as a string conforming to [DNstrings]. For example, it could be "cn=Andrew Rust, ou=Home Office, o=Acme Widgets Inc., c=CA";

4.4 Using SPKM for SECIOP

When the SPKM protocol is chosen as the security mechanism for invoking an object, the SECIOP protocol carries the information described in this section. This protocol is a profile of the SPKM GSS-API mechanism as defined in [SPKMMECH].

All SPKM tokens are encoded according to the general format described in 3.4. The innerContextTokens are described in the following sections. All innerContextTokens are encoded using ASN.1 BER (constrained, in the interests of parsing simplicity, to the DER subset defined in [X.509]).

The SPKM GSS-API mechanism is identified by an OBJECT IDENTIFIER representing "SPKM_1" or "SPKM_2". SPKM_1 uses random numbers for replay detection during context establishment and SPKM_2 uses timestamps (note that for both mechanisms, sequence numbers are used to provide replay and out-of-sequence

SPKM Protocol 4

4.1 Introduction

This chapter specifies the SPKM protocol, a simple public-key GSS-API mechanism. It is based on SPKM as defined by IETF internet draft [SPKMMECH]. SPKM protocol provides CSI level 0 functionality only and the purpose is to allow the adoption of a simple security infrastructure without undue complexity or overhead.

SPKM has two separate GSS-API mechanisms, SPKM_1 and SPKM_2, whose primary difference is that SPKM_2 requires the presence of secure timestamps for the purpose of replay detection during context establishment and SPKM_1 does not. SPKM_1 is the mandatory mechanism for conformance to the SPKM protocol while SPKM_2 is the optional mechanism.

Specifically, it defines the required information for encoding a secure interoperability IOR and defines the token formats used by the SECIOP protocol.

4.2 Cryptographic Profiles

The following cryptographic profiles are supported with this mechanism:

MD5_RSA
Specifies use of the SPKM mechanism to provide data integrity and authenticity by computing an RSA signature on the MD5 hash of that data. The default SPKM key establishment algorithm is used, i.e. the context key is generated by the initiator, encrypted with the RSA public key of the target, and sent to the target. Note that `MD5_RSA` is a mandatory integrity and authenticity algorithm for SPKM.

MD5_DES_CBC
Specifies use of the SPKM mechanism to provide data integrity by encrypting, using DES in CBC mode, the MD5 hash of that data. The default SPKM key establishment algorithm is used.

DES_CBC
Specifies use of the SPKM mechanism to provide data confidentiality by using DES in CBC mode. The default key establishment algorithm is used.

3.4.3 CSI Protocols

This specification includes three protocols for different circumstances as described in 1.2.2 above. In all cases, the appropriate chapter specifies the cryptographic profiles supported, and the contents of the SECIOP security tokens.

In all cases, the protocol as supported by OMG is a subset of the protocol defined in the source document. For example, in all protocols, channel bindings as defined in GSS-API (and specified in the underlying protocols) are not supported. This is because IP addresses cannot be trusted in current implementations; IP addresses are spoofable, therefore including the channel binding information would lead to a false sense of security about the source of the transmission.

The protocols are:

SPKM Protocol

Chapter 4 specifies the SPKM protocol which supports CSI level 0. This is a public key based protocol. The only client information transmitted is its security name.

GSS Kerberos Protocol

Chapter 5 specifies the GSS Kerberos protocol which supports CSI level 1. This is a secret key based protocol. The only client information transmitted is its security name.

CSI-ECMA Protocol

The CSI-ECMA protocol defined in Chapter 6 also supports the privilege handling, separate audit_id and delegation controls of CSI level 2. Sub-schemes within this protocol support the three key distribution options - secret, public and hybrid.

To support this flexibility, the initial_context_token is split into three parts so the attributes for access control are independent of the key distribution method, and this is independent of the cryptography used for message protection. The token contains:

- authorisation information - attributes of a principal are held in a Privilege Attribute Certificate (PAC) with any associated information needed for delegation and other controls. This is independent of the way the communications are protected, so is usable with different key distribution methods.
- security information needed to establish the association. The form of this depends on the key distribution method used. It is a Kerberos ticket if this is secret key based; it is a profile of the SPKM_REQ token for public key mechanisms. In both cases, there is a link between this and the PAC. Changing the security mechanism mainly just requires replacing this part of the token.
- dialogue key packages to establish confidentiality and integrity keys.

TargetResultToken	sent to the initiator by the target to complete establishment of the context in a SECIOP *CompleteEstablishContext* message The token id is 02 00. It is returned by GSS_Accept_sec_context.
ContinueEstablishToken	sent either by the initiator or the target to continue context establishment in a SECIOP *ContinueEstablishContext* message. The token id is 03 00 (in SPKM) It is returned by either the GSS_Init_sec_context call or the GSS_Accept_sec_context call.
ErrorToken	sent on detection of an error during security association establishment in a SECIOP *CompleteEstablishContext* or *ContinueEstablishContext* message. The token id is 03 00 (except in SPKM where it is 04 00). It is returned by either the GSS_Init_sec_context call or the GSS_Accept_sec_context call.

The inner context token for message protection is the message_protection_token in the SECIOP *MessageInContext* message. This can take one of the two following forms:

MICToken	sent either by the initiator or the target to verify the integrity of the user data sent in the following GIOP message (or message fragment). The token id is 01 01 It is returned by GSS_GetMIC.
WrapToken	sent either by the initiator or the target. Encapsulates the input user data (optionally encrypted) along with integrity check values. The token id is 02 01. It is returned by GSS_Wrap.

This specification always use MIC tokens for integrity and Wrap tokens for confidentiality. This may ease national use and export problems where only MIC tokens are supported.

The inner context token in the DiscardContext SECIOP message may optionally contain a DeleteContextToken.

ContextDeleteToken	sent either by the initiator, or the target in a SECIOP *DiscardContext* message to release a Security Association. It is returned by GSS_Delete_sec_context.

- information associated with a principal, including at least an identity. (At CSI level2, there may be a range of privileges and a separate audit identity if required.)
- associated delegation information. Only simple delegation is mandatory to conform to this specification.
- security information used to establish the client-target object security association.
- security information to establish the keys for message protection

3.4.1 Basic Token Format

SECIOP message include context and message protection tokens.

All CSI mechanisms are usable inside and outside the object environment. In line with standard practice outside the object environment, tokens are defined in ASN.1. and encoded for transmission using BER (in some cases, constrained to the DER subset of these). The token appears as a sequence<octet> in CDR encoded SECIOP messages.

These tokens are enclosed within framing as follows:

[APPLICATION 0] IMPLICIT SEQUENCE {

```
    thisMech   MechType
               -- MechType is OBJECT IDENTIFIER
    innerContextToken ANY DEFINED BY thisMech
               -- contents mechanism-specific;
    }
```

[Note 1: For conformance to GSS-API, only the initial context token need use this token framing. However, in the CSI protocols, it applies to all tokens.

Note 2: CORBASEC says that the initial context token should include a mechanism version as well as type. For CSI mechanisms, version numbers are in the mechanism specific information such as the Kerberos ticket or CSI-ECMA PAC.]

3.4.2 Inner Context Tokens

The same token types are used in the different CSI protocols, though not all protocols support all token types. The token types are defined below showing the relationship with GSS-API calls, as all CSI protocols can all be implemented using GSS-API.

The inner context tokens used for security association establishment are:

InitialContextToken	sent by the initiator to a target, to start establishment of a securityassociation in a SECIOP *EstablishContext* message. The token id is 01 00 (hex). If GSS-API is being used, it is the value returned by the GSS_Init_sec_context call.

In all cases, support of a CSI protocol requires support for a cryptographic profile which provides integrity of user data, but not confidentiality, as such as profile is easier to deploy internationally. For example, the GSS Kerberos protocol always supports its *MD5* cryptographic profile. Other profiles may also be supported.

3.3.4 Security Name

The form of the security name depends on the security mechanism used (see chapters 4, 5, and 6 for details) For example, it can be a Kerberos name or a Directory style name. Directory names conform to the string representation defined in [DNstrings].

The security name may be at the component level of the IOR or higher if shared between mechanisms. If a security mechanism tag, but no security name is present in the IOR, the IOR is improperly formatted and an INV_OBJREF exception shall be raised when the IOR is used to specify the target of an operation.

3.3.5 Security Administration Domains

As defined in CORBASEC, a security policy domain is a set of objects to which a security policy applies for a set of security related activities and is administered by a security authority.

Security mechanisms are concerned with the security domains where users and other principals are administered, often by on-line authorities such as Authentication and Privilege Attribute Services. This domain will often be the enclosing domain encompassing secure invocation, access control and other policy domains.

Note that some authorities may be off-line. For example, the Certification Authority used to issue certificates is often off-line.

The security mechanisms specified in this document allow requests to cross domain boundaries. At the boundary, trust between the domains needs to be established. (The way this is done depends on the mechanism used.) Also, the scope of privileges may not always cross the domain boundary. This specification does not define how privileges are mapped on crossing domain boundaries, as this does not affect the protocol.

While all security mechanisms here include the concept of such domains, in Kerberos (used here as the secret key mechanism) these are known as realms. So in this specification, in tokens using this mechanism, the term realm is used.

3.4 SECIOP Protocol

The SECIOP protocol includes the tokens for context establishment and management and per message tokens.

The context establishment tokens contain:

The security mechanisms defined here allow a choice of algorithms which can be used for these different functions depending on the needs of the functions and also the requirements for international deployment in countries which constrain how cryptography can be used and exported from countries where use of cryptography is controlled. In some cases, export controls may require international versions of products to use shorter key lengths. Therefore a potentially large number of combinations of algorithms and key lengths are possible.

However, for interoperability, both client and target must support the same algorithms and key lengths for these functions.

This specification defines a number of *cryptographic profiles*, where each profile identifies a set of algorithms with specified key lengths used by a mechanism for specified functions.

For example, the CSI-ECMA protocol defines a *NoDataConfidentiality* cryptographic profile which can use DES and RSA for protecting the security mechanism, but does not encrypt the ORB request/reply. (The profile for full security would use DES/64 for data confidentiality.)

Cryptographic profiles are identified by a value, represented in IORs as an unsigned short i.e.:

```
typedef    unsigned short    CryptographicProfile;
```

Key Establishment Algorithms

The algorithms used to establish the cryptographic session keys during security associations depend on the type of mechanism. Where the secret key (Kerberos based) mechanism is used, either via the GSS Kerberos or CSI-ECMA protocol, the DES algorithm is used. When a public key mechanism is used, either via SPKM or CSI-ECMA protocol, the RSA algorithm is used.

Common Message Protection Algorithms

Even if different mechanisms and algorithms are used for key establishment, the same algorithms can be used for message protection.

All CSI mechanisms have cryptographic profiles which include an MD5 hash of the data for integrity, though the hash, in some profiles may be signed or encrypted.

All CSI mechanisms can use DES in CBC mode for message confidentiality.

Cryptographic profiles supported by CSI protocols

A number of cryptographic profiles are defined for each CSI protocol. (Further cryptographic profiles using different algorithms can be used with these protocols, but these are not part of this interoperability standard.) A target may support several cryptographic profiles for a particular mechanism.

Tag ids for the mechanisms are:

```
TAG_SPKM_1_SEC_MECH
TAG_SPKM_2_SEC_MECH
TAG_KerberosV5_SEC_MECH
TAG_CSI_ECMA_Secret_SEC_MECH
TAG_CSI_ECMA_Hybrid_SEC_MECH
TAG_CSI_ECMA_Public_SEC_MECH
```

The association options required/supported by the target are defined in 3.3.2 below.

The sequence of crypto_profiles defines one or more cryptographic profile supported by this target using this mechanism as defined in 3.3.3 below.

The security name is defined in 3.3.4 below.

3.3.2 Association Options

With all CSI protocols and mechanisms, a secure ORB supporting a target object must be able to put in the IOR any or all of the association options defined in the CORBA security specification into the IOR, as required by the target.

All compliant secure ORBs supporting clients must be able to accept all the `target_supports` and `target_requires` association options, and act on these correctly as defined in CORBASEC.

However, two of the association options are replay and misordering detection. While all the protocols in this specification include facilities to detect replay and misordering, in a multithreading CORBA environment, the calls on the security mechanism are not guaranteed to be made in the same order that the messages they are protecting are transmitted. Therefore the facilities in the security mechanisms cannot guarantee they will correctly detect replay and misordering. An extension to SECIOP is expected in future to provide these checks - see B.3.2. Until this change to SECIOP has been specified and adopted, although these association options may be set, replay and misordering detection is not a mandatory part of this specification.

If no association options are specified in the IOR, a CSI defined default is assumed.

3.3.3 Cryptographic Profiles

Cryptographic algorithms are used for:

- integrity and confidentiality protection of messages
- establishing the security association between client and target (including peer authentication and establishing session keys)
- deriving dialogue keys for message protection (both confidentiality and integrity)
- protecting systems security data such as PACs (Privilege Attribute Certificates)

3.3 IOR

The IOR TAG_INTERNET_IOP profile contains the security tags needed for common secure interoperability using GIOP/IIOP. These security tags may be shared with other (non IIOP) protocols, including DCE-CIOP.

The security tags describe what the security the target supports and requires and any mechanism specific data required for secure interoperability using this mechanism. (Security tags are described in Chapter 8 of the CORBA Security specification.)

For common secure interoperability, for all CSI mechanisms and protocols, the IOR must contain at least one appropriate TAG_x_SEC_MECH tag.

The IOR may also contain the following tags as defined in Chapter 8 of CORBASEC:

- a TAG_SEC_NAME, which provides the security name and may be shared between mechanisms which use the same form of name. Conformant implementation must be able to accept security names shared between such mechanisms.
- a TAG_ASSOCIATION_OPTIONS which may be shared between mechanisms
- TAG_GENERIC_SEC_MECH whose component definition includes a `sequence <TaggedComponents>` which includes a security_mechanism_type and can include a security name and association options.

If a mechanism is selected for use, and has a defined security name and/or association options, these are used in preference to any values for these defined at the higher level. If no name or no association options are defined for the mechanism, then the values of these tags in the IIOP profile are used.

3.3.1 Mechanism Tags

The TAG_x_SEC_MECH tags for all the CSI mechanisms defined in this specification have an associated component data structure of the same form:

```
struct <mechanism name>   {
    AssociationOptions               target_supports;
    AssociationOptions               target_requires;
    sequence <CryptographicProfile>  crypto_profiles;
    sequence <octet>                 security_name
};
```

The mechanism names for the CSI mechanisms are:

```
SPKM_1
SPKM_2
KerberosV5
CSI_ECMA_Secret
CSI_ECMA_Hybrid
CSI_ECMA_Public
```

- *MessageInContext*: used to sent messages representing the object requests and responses within the context, once this has been established. It includes a *message_protection_token*. This provides integrity and/or confidentiality of the message in transit.

The message headers for all these messages are defined in the CORBA Security specification, but the content of the security tokens exchanged are dependent on the security mechanism and for the CSI protocols are defined in this specification.

The *initial_context_token* is sent from the client to the target object to establish the security association. In addition to this initial token, subsequent context establishment security tokens may be needed, for example, if mutual authentication of client and target is required, or some negotiation of security options for this mechanism is required, for example, the choice of cryptographic algorithms. This CSI specification does not include mechanism negotiation, as all required information can be carried in the IOR. (If negotiation were included, this could decrease the IOR size at the expense of extra protocol exchanges).

Note: some revisions to SECIOP as defined in the 95-12-1 version of CORBASEC are assumed in this specification. These are in line with the revision to CORBASEC being produced at the time of writing this specification - see Appendix B.3.

3.2 Introduction to the Common Interoperability Protocols

All the CSI protocols and mechanisms use common elements as far as possible.

- All mechanisms use IOR tags of the form TAG_x_SEC_MECH as defined in CORBASEC section 8.4.
- The component data structure associated with these tags is common for all protocols and mechanisms in this specification.
- Cryptographic profiles are defined in all cases which allow use of relevant algorithms for confidentiality, integrity etc. Different mechanisms support some of the same algorithms and one way functions.
- The MechanismType as seen at the IDL interface also reflect the mechanism ids and cryptographic profile values in the IOR tags.
- Privilege attributes when CSI level 2 is used, are the same whether a secret or public key mechanism is used.
- The basic SECIOP token format and some details (such as token types and ids) is common for all protocols.

Note: datatypes in the Security and SECIOP modules defined in the CORBA Security specification are referred to from this specification. These include, for example, `AssociationOptions, SecurityName, MechanismType`.

These protocols are designed to allow use of GSS-API mechanisms. However, use of level 2 facilities such as handling of privileges, as defined in CORBASEC, imply use of an extended GSS-API such as [XGSSAPI].

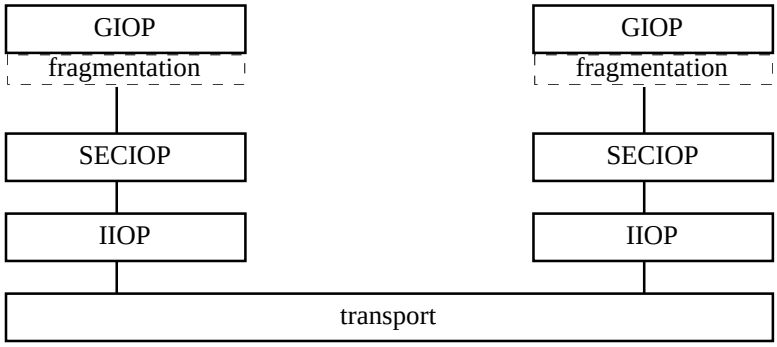
- a security name or names for the target so the client can authenticate its identity.
- any security policy attributes of the target relevant to a client wishing to invoke it. This covers, for example, the required quality of protection for messages.
- identification of the security mechanism(s) supported for secure communication and any associated mechanism specific data. This allow the client to use the right security mechanism and cryptographic algorithms to communicate with the target.

This specification defines details of the security mechanism tags in the IOR for the common secure interoperability mechanisms and associated information specified here.

3.1.2 Client - Target Protocol

The protocol between client and target object on object invocations establishes a "secure association" between the client and target (if there is not already one) by transmitting security token(s) between them transparently to the application.

When using the standard CORBA 2 GIOP/IIOP protocol, the security tokens needed to establish and control the secure associations and the protected messages are part of the Secure Inter-ORB Protocol (SECIOP). This protocol sits below the GIOP protocol and provides a means of transmitting GIOP messages (or message fragments) securely.



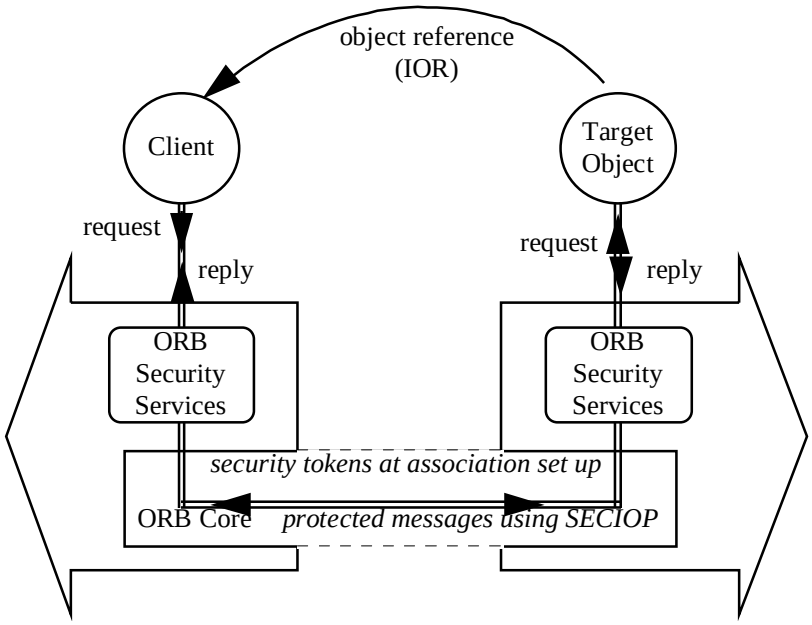
SECIOP defines the following message types:

- *EstablishContext*: passed by the client to the target when a new association is to be established. This includes an *initial_context_token.*
- *CompleteEstablishContext*: returned by the target to indicate the association has been established. This includes a *final_context_token*.
- *ContinueEstablishContext*: passed by the client or target during context establishment to pass further messages to its peer as part of establishing the context. If present, it includes a *continuation_context_token*.
- *DiscardContext*: used to indicate to the receiver that the sender of the message has discarded the identified context. This optionally includes a *delete_context_token*.
- *MessageError*: used to indicate an error detected in attempting to establish an association or errors in the use of the context.

Common Mechanism Information 3

3.1 CORBA Security Interoperability

The Common Interoperability protocols conform to the CORBASEC model for secure interoperability using the CORBA 2 interoperability standard GIOP/IIOP protocol is shown in the following diagram.



3.1.1 Object Reference

When the target object registers its object reference in a secure environment, this contains extra security information to assist clients in communicating securely with it. The CORBA Security specification (chapter 8) specifies TAGs to go in the CORBA 2 Interoperable Object Reference (IOR) for the following security information:

- Mapping of attributes as described in 2.4 above affects replacable security policies which use these attributes.
- Use of the Generic Security Services API (GSS-API) within the Vault and Security Context implementation objects defined in CORBASEC should make these objects independent of the particular security mechanisms used

Attribute Mapping

As described in 2.4.3 above, the form of attributes may need to be mapped before being made available to a target security policy (AccessPolicy or AuditPolicy) or to the target object.

Currently CORBASEC does not specify an interface to an attribute mapper, so it is not possible to replace attribute mapping independently of the ORB/security mechanism

In future, an extension to the CORBA Security specification may be proposed to allow this attribute mapper to be replaced - see Appendix B section B.3.3.

Use of GSS-API

The choice of security mechanism is not visible outside the Vault and Security Context objects, except for the identification of the Mechanism (and associated cryptographic profiles) in the IOR and in response to `get_mechanism` and similar operations.

The Vault and Security Context can themselves use GSS-API to implement their security functions, and so remain independent of security mechanism.

If only CSI level 0 or 1 facilities are used, the standard GSS-API interface (as defined in RFC 1508) can be used. If CSI level 2 facilities are required, this requires use of attributes other than the security name, and may also use delegation controls. It therefore requires use of an extended GSS-API, such as [XGSSAPI].

Use of GSS-API is a recommendation, but is not proposed as a conformance option in for this CSI specification or for the CORBA Security specification.

2.5.3 Delegation Related Interfaces

Interfaces to handle no delegation, simple delegation and composite delegation (hence delegation interfaces for CSI levels 0, 1 and part of 2) are already defined in CORBASEC.

CSI level 2 also supports controls on the delegation of credentials. The way of specifying these controls is not included in this, or the CORBASEC specification. It is assumed to be done by administrative action. For example, it may be done by associating the delegation controls with a user or an attribute set selected when the user logs on or selects attributes at other times. In line with CORBASEC, management of attributes associated with a principal is considered out-of-scope of this specification.

No facilities are currently defined for an application object to specify controls it wishes to apply on delegating its credentials. In future, such facilities may be considered for CORBASEC - see CORBASEC Appendix G section G.10.

2.6 Support for CORBASEC Facilities and Extensibility

This CSI specification assumes that the ORB conforms to at least CORBA Security mandatory facilities (except for delegation at CSI level 0), and requires that this functionality can be supported across different ORBs using any of the CSI conformance points specified here.

The CORBA Security specification allows use of a wide range of security policies, facilities and mechanisms. ORBs conformant to this CSI specification can restrict which of these can be used during interoperability in the following ways:

- the protocol may not carry the privileges the target needs for some of its access policies. For example, at CSI levels 0 and 1, only an identity is supported.
- it may not carry the type of audit identity needed for the audit policy, for example, it may not be able to carry an anonymous audit_id.
- it may not support composite delegation. (CSI levels 0 and 1 do not; in CSI level 2 it is not mandatory)
- there are restrictions on the SECIOP exchanges e.g. separate request and response protection is not supported
- unauthenticated users may not be supported (All CSI levels)

2.7 Security Replaceability for ORB Security Implementors

CORBASEC defined how security policy implementations could be replaced to provide new security policies, for example, access policies, independently of the particular ORB used, provided it supported the replacability conformance option.

This common interoperbility specification affects replacability in two areas:

2.5.2 Mechanism Types

In the CORBA Security specification, the mechanism at the application interface is defined as `Security:MechanismType` (a string). For the CSI mechanisms, this specifies the mechanism and zero, one or more cryptographic profiles separated by commas.

The mechanisms supported by an object are identified by tags in its IOR. In the MechanismType, the mechanism is identified by a "stringified" form of the TAG_x_SEC_MECH id value for that mechanism. Mechanisms supported by CSI protocols are:

- *SPKM_1* or *SPKM_2*: the level 0 public key mechanisms using the SPKM protocol
- *KerberosV5:* the level 1 secret key mechanism using GSS Kerberos protocol
- *CSI_ECMA_Secret*: the CSI-ECMA secret key mechanism, using Kerberos V5
- *CSI_ECMA_Hybrid*: the CSI-ECMA mechanisms which uses secret key technology for key distribution within a domain, but public key between domains
- *CSI_ECMA_Public*: the CSI-ECMA public key mechanism

Cryptographic profiles are identified by a "stringified" form of the (unsigned short) CryptographicProfile value as used in the IOR.

MechanismType is used in a number of CORBASEC operations. These include:

- operations which obtain the mechanisms and cryptographic profiles available such as `get_security_mechanism` on an object reference.
  In this case, the MechanismType contains all the Cryptographic profiles available with that mechanism to communicate with that target.
- operations which specify a security mechanism to use when talking to a target e.g. `override_default_mechanism` on an object reference and `init_security_context` on the Vault
  In this case, just the mechanism name may be specified (in which case, a default cryptographic profile will be used) or a mechanism name and cryptographic profile may be specified.

The cryptographic profiles allowed with each mechanism are defined in the appropriate chapter for that protocol.

The `get_service_information` operation on the ORB can also return the mechanism, though in this case, it is in the form of a `sequence<octet>`.

This specification also uses mechanism tags in the IOR and mechanism type Object Identifiers (as in GSS-API) in SECIOP messages (see chapter 3).

Mapping to Local Attribute Values

An ORB can support mapping of the security name and other attributes to local operating system values such as UNIX uids and gids. This mapper could generate different access_ and audit_ids. Note that when using local values, the application (particularly the access policy administration) will not be portable to other types of system.

The way mapping of these values is done is specific to the ORB and/or operating system - this standard does not specify rules about how this mapping is done, whether it calls on other software to do it, and what types of values it generates.

However, the defining authority in the IDL SecurityAttribute must identify the local environment responsible for the meanings of these values, so the application can determine where these values are valid.

Mapping to local attributes may be done by an optional attribute mapper - see 2.6.1.

2.5 CORBA Interfaces

This specification:

- extends the `get_service_information` operation on the ORB defined in the CORBA security specification to add common secure interoperability options
- defines profiles of the interfaces there, to specify values of some parameters
- specifies restrictions which apply to the application when conforming to this Common Secure Interoperability standard

2.5.1 Finding Security Features

CORBASEC defined a new operation on the ORB to `get_service_information`. For the CSI standard, extra ServiceInformation is returned when the ServiceType is Security.

Three new Service Options are added:

```
const   ServiceOption      CommonInteroperabilityLevel0 = 10;
const   ServiceOption      CommonInteroperabilityLevel1 = 11;
const   ServiceOption      CommonInteroperabilityLevel2 = 12;
```

The common interoperability protocols supported are identified using a ServiceDetail structure with a ServiceDetailType of SecurityMechanismType, as defined in CORBASEC. The values for the CSI mechanisms are defined in 2.5.2 below.

The security name when using a public key based mechanism is a directory name. This is a multi-part name e.g. country, organisation, organisation unit, surname and common name. This is returned from the security mechanism in the form of a string complying with [DNstrings] for the string representation of distinguished names. The separators between components of the name may be commas or semicolons.

In both cases, the full Security name is used as the value for the access_id and audit_id in the IDL SecurityAttributes. Note that this means the form of these attributes are dependent on the security mechanism used, as Kerberos and X.500 names have different forms.

Mapping other Attributes to Externally Valid IDL Attributes

Other security attributes may also be transmitted from the client when using the CSI-ECMA protocol. For example, at level 2, there could be a role, groups and enterprise specific attributes as well as access_id and/or audit_id. Also, separate access_ and audit_ids may be transmitted.

These in general will already have values which are valid outside a particular ORB and operating system. So the mapping is mainly to put these in the form of an IDL SecurityAttribute. However, if a separate audit_id has not been transmitted, the audit_id value will be copied from the access_id. Also, if a separate defining authority is not transmitted for an attribute, the defining authority for the attribute in IDL is set from the issuerDomain of the authority who generated the Privilege Attribute Certificate containing the privileges. Note also that the target security policy may restrict which of the attributes are available to the application.

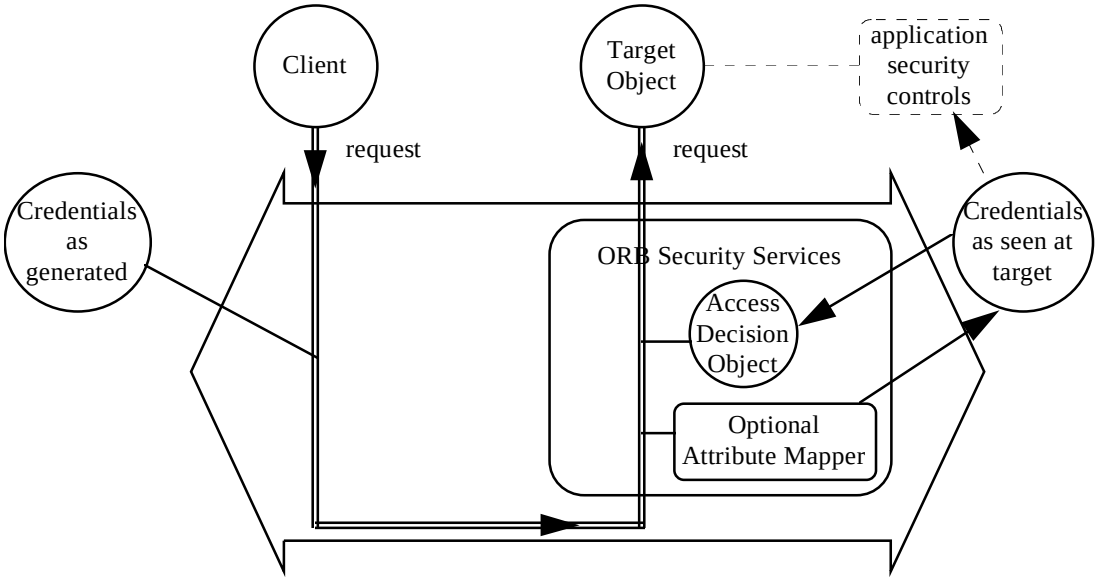
Attribute types in transmission are identified by Object Identifiers. For the standard attribute types such as role, group (as defined in Appendix A of CORBASEC), the type is automatically translated to the appropriate CORBA family and attribute type. The value is also re-encoded, if needed, from ASN.1 to the equivalent IDL type.

We propose that OMG should register itself in the ISO Object Identifier space. Then a SecurityAttribute type where there is a family definer registered with OMG (see CORBASEC Appendix A.9) can be transmitted with an Object Identifier of:

```
<iso>..<omg>.<security>.<family definer>.<family>.<attribute type>
```

This can then be mapped automatically onto the CORBA SecurityAttribute structure.

Attributes other than the standard ones and those with CORBA family Object Identifiers are not guaranteed to be understood at the target, so may not be automatically mapped to CORBA families and types. Such mapping can be done by an optional attribute mapper which understands these attribute types.



This mapping depends on:

- which functionality level is supported. At levels 0 and 1, a single name must be mapped to provide both access_id and audit_id. This will be the security name if the protocol does not carry a separate access_id or audit_id; both the SPKM and GSS-Kerberos protocols use the security name.
- whether the access control decisions at the target uses attribute values which are valid externally from the ORB/operating system (for example, in a domain of heterogeneous systems), or whether the Access policies uses local attributes (such as operating system ids).

  In line with the OMG requirement for portability, externally valid attributes are the norm, and must be supported in conformant ORBs (so that an application which includes administration of its access policy is portable between unlike systems). Mapping to local attributes may also be provided, but is not standardised in this specification.

Mapping Security Names to Externally Valid Identities

Where the only client attribute transmitted is the security name, CSI conformant ORBs map this onto both the access_id and audit_id in the received credentials. These two both have the same value.

The security name when using the GSS-Kerberos protocol has two components, a realm name and a principal name. The security name is of the `principal@realm`. The principal name may be a multi-component name with components separated by slash (/) - see [GSSKRB5] section 2.1.1.

2.4.2 Attributes During Transmission

At levels 0 and 1, only the principal's identity is transmitted, no other attributes.

At level 2, a wide range of privileges can be transmitted including standard CORBA ones and optionally user defined ones. Attributes may have individual defining authorities, as at the IDL interface, or share a defining authority.

2.4.3 Attributes at the Target

At CSI levels 0 and 1, when only a single identity (e.g. the security name) is transmitted, this is used to generate the access_id and the audit_id at the target. (Note that when using the CSI-ECMA protocol at level 0 or 1, principal identity attributes are transmitted separately from the security name, so the access_id and audit_id do not have to be generated from the security name.)

At CSI level 2, all conformant ORBs can accept:

- separate access and audit ids or a single identity used for both purposes.
- transmission of any privileges defined in the CORBA security specification and any privileges with Object Identifiers which can be mapped to CORBA SecurityAttributes.

This range of privileges can be used in access decisions at the target. Even if these privileges are not used by the invocation access policy to control access to the target object, they may be obtained by the application using `get_attributes` and used in application access decisions.

The attributes at the target appear as defined in CORBASEC. i.e. they have:

- an Attribute type (family definer, family and the type within this family)
- a defining authority
- the attribute value

The attributes may need to be mapped from their form in transit, to the form used at the IDL interface in response to `get_attribute` calls. So an attribute mapper may be needed as shown in the following diagram.

`get_attributes` function. It could then call in a non-standard way on whatever service provides privileges in this case. Alternatively, an attribute Mapper (see 2.4.3) could be used before calling the access policy (if this optional facility is supported).

Audit policies generally require an audit id, though this may be derived, like the access id, from a single identifier.

The CORBA Security specification allows unauthenticated and authenticated users. However, unauthenticated principals do not have identity attributes or privilege attributes. In the protocols defined here, principals must be authenticated.

The privilege and other attributes as seen by the Access Decision functions at the target may not be those passed from the client as the security mechanism may have moderated what is made available to the object system.

2.4.1 Credential Content at the Client

Credentials are made available to the client as the result of authenticating the user (or other principal), though they may be modified later. Authenticated users have two types of attributes visible to applications and relevant to secure interoperability:

- privilege attributes used for access control. These include the access_id (the principal's identity as used for access control), other CORBASEC defined attributes such as groups, roles, security clearance, and enterprise defined attributes
- identity attributes used for purposes other than access control. Only the audit identity is relevant here.

At CSI levels 0 and 1, the only attributes which must be visible to the client and target are the access-id and audit_id. (These will normally be the user's security name - see 2.3.3 below).

At CSI level 2, a wider range of privilege attributes is supported.

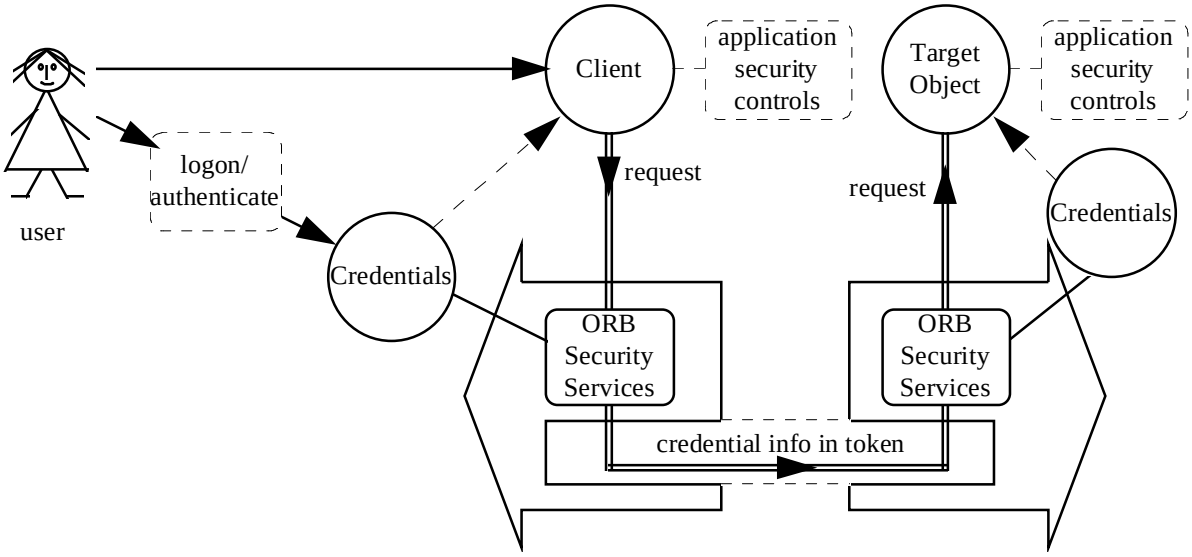
- all conformant ORBs can generate (via security services) credentials with the following privilege attributes. (For the definition of these, see CORBASEC):
  - access_id
  - audit_id
  - role
  - groups - a primary group and other groups
- there may be a single identity (e.g. the access identity) which can also be used for auditing, or separate access_ids and audit_ids may be generated. Audit_ids may be anonymous.
- there may optionally also be other privilege attributes including user defined ones.

Security Policies

Security policies are potentially sharable between ORBs if they use only identities and privileges which are available at both ORBs and can be transmitted between them. For example, a DomainAccessPolicy which uses roles must receive requests from an ORB which can generate them via a CSI level 2 protocol which can transmit roles.

2.4 Model for Use and Contents of Credentials

The CORBA Security model includes security functionality enforced during object invocations and by applications as shown in the following diagram.



Most of the security services utilise the principal's credentials either at the client, before invoking the target object or at the target. For example, the ORB security services use these credentials for secure associations, access control and auditing.

To fit with the standard CSI security mechanisms, user/principal authentication must produce credentials suitable for both client side security controls and to fit with the security mechanisms used for secure invocations. A single credentials object may have security context information for more than one mechanism.

Security services at the client application use these credentials to enforce security there.

Access control policies at the target generally depend on the initiating principal's privilege attributes (which generally includes an identity). They therefore normally rely on information from the credentials being passed from the client to the target. Other access policies may use the pull model for obtaining privileges at the target. For example, an access policy at the target could obtain the access identity using the

CORBA attribute family definer, and defines its own families of attributes. However, some attribute types defined outside the object system may not be understood at all targets, so portability of these may not be possible to all environments.

Audit

Auditing is as defined in CORBASEC and is possible at all CSI levels. A separate audit_id (which may be anonymous) can be transmitted at CSI level 2.

Secure Invocation

Conformant implementations (all CSI levels) must support all the association options defined in CORBASEC.

Channel bindings, as defined in GSS-API and all protocols defined here, are not part of the mandatory specification.

Conformant implementations at level 2 allow use of algorithms with different strengths for integrity and confidentiality.

Delegation Facilities

At CSI level 0, no delegation is supported.

At CSI level 1, the initiating principal's identity can be delegated to the target. It is either delegated or not - there are no other restrictions on delegation.

At CSI level 2, the initiating principal's privileges as well as identity can be delegated to the target. Delegation can be controlled further - restricting the targets to which the attributes can be delegated. These restrictions must be specified by administrative action, as there are no security interfaces specified in CORBASEC to do this.

Level 2 protocols are also defined which allow support of composite delegation. However, support of this is not required by conformant ORBs.

Non Repudiation

Non-repudiation relies on NR credentials for handling NR evidence tokens. CORBASEC allows the same credentials to be used for secure invocations and non-repudiation. This will only be possible if compatible security technology is used for non-repudiation and secure invocation. While no specific security technology is mandated for non-repudiation, it is expected that this will use public key technology. So common credentials usable for both purposes are expected to use public key technology, so fit with public key mechanisms (SPKM or the CSI-ECMA public key option), rather than with secret key mechanisms.

In this example, Bob wants to close his bank account and is prepared to give Dan power of attorney to do this.

- At CSI level 0, no delegation is possible, so Bob has to go to the bank and close the account himself
- At CSI level 1, Bob gives Dan unlimited power of attorney to act as him (as delegation is unrestricted). Dan can close Bob's bank account.
  As the power of attorney is unlimited, Dan can also read Bob's medical records and pass on the power of attorney to Mark - who can also close Bob's bank account, read Bob's medical records etc
- At CSI level 2, Bob gives Dan the power of attorney to close his bank account, so Dan can close the account. But this does not include the right to read Bob's medical records (as only limited privileges were given to Dan) and does not include the right to give the power of attorney to Mark (as delegation was restricted to Dan)

2.3 Security Functionality

This section reviews the security functionality in CORBASEC and specifies which functionality is supported interoperably at which CSI level. Some security functionality is supported at all CSI levels, some only at CSI level 1 or 2.

Authentication

The CSI mechanisms do not specify authentication of principals, though use the result of such authentication. So principal authentication must result in credentials which contain the security information needed by the security mechanisms supported by this conformant ORB.

CSI mechanisms require authenticated principals. *[See CORBASEC 3.3]*

Access Control

Access controls depend on the privileges of the principal.

At CSI levels 0 and 1, only the principal's identity is available at the target. So Access Policies using this level must either:

- use only the principal's identity for access control
- retrieve other attributes for that principal prior to taking the access decision (the "pull" model).

The standard DomainAccessPolicy assumes all privileges required have been "pushed" from the client, so will be restricted to using identity only. Access policies using the pull model will not be portable, if the source of such attributes is system dependent.

At CSI level 2, the AccessPolicies can use any of the privileges supported by both ORBs. All CSI level 2 conformant ORBs support access_id, groups and roles. They may also transmit user defined privileges, where the user enterprise concerned has a

2.2.2 Common Secure Interoperability Level 1

CSI level 1 supports identity based policies with unrestricted delegation.It requires ORBs to support the mandatory part of the CORBA Security when two conformant ORBs interoperate (using the same security mechanism). It provides the CSI level 0 facilities listed in 2.2.1 plus:

- security information, in particular, the security name, of a principal in the call chain can be delegated to encapsulated objects (subject to security policy).

  Once this security information has been delegated, the intermediate object has the choice of acting under its own identity or delegating the initiating principal's identity when invoking another object. When delegating another principal's identity, the delegated identity (rather than the immediate invoker's identity) is used to set both the access_id and audit_id at the target.

2.2.3 Common Secure Interoperability Level 2

CSI level 2 supports identity and privilege based policies with controlled delegation. ORBs supporting this level must support interoperability of all facilities in the CORBA Security specification concerned with object invocation. CSI level 2 provides the CSI level 0 and level 1 facilities listed above plus:

- the security information of the immediate invoker or the delegated information of the initiating principal can include more security attributes as follows:
  - an extensible range of privilege attributes e.g. roles, groups, enterprise defined attributes, so supporting a wider range of policies. These generally include an access_id which is independent of the security name (and hence the mechanism type used) and is used to set the access_id at the target. Interoperability using particular types of privileges depends on these being common to both ORBs. This CSI specification defines which privileges a CSI level 2 conformant ORB must support - see 2.2
  - a separate audit_id can be transmitted. This may be anonymous (except to the audit administrator). It will always represent the actual principal using the system, evem when the access_id represents someone who has allowed another user to access the system on his behalf.
- the delegation of a prinicipal's attributes can be controlled - for example, usable at only identified (groups of) targets. So an intermediate receiving delegated security attributes of a principal will not always be able to delegate them.
- composite delegation is allowed for, though support for this is not mandatory.

2.2.4 Example

This section looks at an example of a secure object system which highlights the difference between the delegation facilities of the three CSI levels.

Note that the interoperability defined here is for interoperability of requests/responses between ORBs. It does not include interoperability of the evidence tokens used for non-repudiation.

2.1.2 Replaceability

CORBASEC defines replaceability options to allow ORB implementors to support a wide range of security policies and mechanisms.

For example, the standard DomainAccessPolices can be replaced by other policies where ORBs support the appropriate CORBASEC replacability option. This specification still allows this replacability, though the policy being added may be restricted by the security information guaranteed to be available.

Also, a further replacability point is proposed to provide optional mapping of attributes received onto those used in access control decisions at a particular target.

CORBASEC allows replacability of security mechanisms by replacement of the Vault and Security Contexts objects. This specification defines mechanisms and protocols which can be implemented via a GSS-API interface. This adds the potential for having a single implementation of the Vault and Security Context objects, which by using GSS-API, should be able to use different security mechanisms.

2.2 Interoperability Levels

This specification includes three interoperability levels as outlined in 1.2.1 above. This section gives more information about these levels and an example showing the difference in the way in the way they handle a particular problem.

2.2.1 Common Secure Interoperability Level 0

CSI level 0 supports identity based policies without delegation. It requires ORBs to support the mandatory part of CORBA Security specification when they interoperate (using the same security mechanism) except that delegation need not be supported. The following are supported:

- authentication of principals using security functions under one ORB and then use of the resultant credentials when making a secure invocation to an object under a different ORB
- secure associations to establish trust between client and target and protect messages
- as part of the secure association, the security name of the client is passed to the target and used to set both access_id and audit_id so that identity based access and audit policies can be supported.

  Note, however, that the identity is always that of the immediate invoker of an object - in a chain of object invocations, this is only the same as the initiator of the chain at the point of entry to the chain.

Security Facilities and Interfaces 2

2.1 Introduction

This chapter defines the effect on the security facilities and interfaces defined in the CORBA Security specifications when using the Common Secure Interoperability standards specified in this document. It is aimed at:

- object implementors developing applications using a secure object system who need to know what security is available
- implementors of security policies who may be constrained by the security attributes available when interoperating according to this standard.
- ORB implementors supporting replaceable security policies

Information required by security implementors to implement the security mechanisms is in chapters 3 onwards.

2.1.1 Functionality

The CORBA Security specification [CORBASEC] defines security functionality available to secure object systems both for applications which are unaware of security and for those which want to enforce security policies themselves. CORBASEC is designed to allow a range of security policies to be used. It does define some standard policies, for example, the DomainAccessPolicy, but even in this case, it does not constrain, for example, the types of privileges used in access decisions.

When ORBs interoperate, an application may be distributed over several ORBs, not all of which support the same security facilities, and are therefore capable of supporting the same security policies.

This common secure interoperability specification defines what security information is transmitted between ORBs, and therefore what security facilities and policies are supported in an interoperable environment. It defines two levels of functionality and is more precise than CORBASEC in specifying the particular security attributes conformant ORBs must support.

Chapter 5 defines details of the GSS-Kerberos protocol which provides CSI levels 0 and 1 functionality using secret key technology.

Chapter 6 defines the CSI-ECMA security protocol which supports CSI levels 0, 1 and 2 using secret, public key and hybrid mechanisms. It includes subsections on the particular mechanisms which are part of this specification, and is extensible, so could also be used with other security mechanisms which do not form part of this specification.

Appendix A gives the complete protocol IDL specification for all conformance points.

Appendix B describes changes required to the CORBA core and CORBA Security specification.

Appendix C describes potential secure interoperability options not included in this specification.

Appendix D lists the main documents which this specification refers to.

Chapter 2 is relevant for readers implementing objects in a secure, interoperable environment. It also has some information for ORB implementors which is independent of security mechanisms or protocol.

Chapters 3 to 6 are aimed at implementors of the security mechanisms.

1.7 Proof of Concept

This specification defines three functionality levels for which an ORB may provide secure interoperability and three protocols - SPKM, GSS Kerberos and CSI-ECMA.

The SPKM protocol defined here is currently available in Entrust implementations from Nortel.

The Kerberos protocol defined here includes an enhancement (mainly for delegation) of the beta 5 MIT Kerberos V5 implementation. The delegation enhancement has been implemented in the recently released MIT Kerberos beta 6, though that is not yet fully compliant with [GSSKRB5].

The CSI-ECMA protocol defined here is a minor variant of that in the current SESAME V4 implementation used in commercial products from SESAME partners. This supports all CSI functionality levels defined in this document.

Other parts of this specification define details or profiles of the CORBA Security specification, so are covered by the proof of concept statement there.

- the Kerberos V5 technology is licensable from the Massachusetts Institute of Technology without cost and is widely deployed within the USA. However, it is subject to export control from the USA. Therefore, [GSSKRB5] is the definition of the protocol used here, as this can be implemented independently of the MIT Kerberos code.
- SPKM implementations are available, though not free. As for other mechanisms, the (draft) standard is the basis of this specification
- SESAME implementation is available, but is not free for commercial use, and has restrictions on cryptography for export reasons (the public version does not include commercial cryptographic profiles - it has the secret key algorithm replaced by XOR for export control reasons)
- There are two patents associated with the CSI-ECMA protocol. These will be usable free of charge for implementations conformant with this specification under fair conditions (Formal definition of these are available from Bull and ICL).
- the DES algorithm is widely deployed internationally, but is subject to export controls. Export with key lengths which provide strong confidentiality is not generally permitted.
- the RSA algorithm is increasingly widely deployed internationally. However, it is subject to licensing in the USA. It is also subject to export controls, though where it can be shown that it is not used for confidentiality, products using it are more likely to be exportable.
- any other cryptographic algorithms used are generally subject to export controls, as is any interface which makes it easy to replace algorithms.

Identifying Changes to SECIOP

Some revisions required to the SECIOP protocol as defined in the CORBA Security specification have been identified during the production of this specification - see Appendix B.

1.6 Specification Structure and Readership

Chapter 1 introduces the specification and gives an outline of the facilities specified. It also describes the requirements which led to this choice of facilities.

Chapter 2 defines the security facilities guaranteed to be usable when interoperating between secure ORBs which conform to this specification. It distinguishes facilities provided at different CSI levels.

Chapter 3 defines the elements of the protocols which are common across CSI protocols. This includes the IOR as well as security tokens in SECIOP.

Chapter 4 defines details of the SPKM protocol which provides CSI level 0 functionality using public key technology.

Applications should be unaware of the security mechanism used to enforce the security, unless they specifically ask what it is e.g. using `get_service_information` (see CORBASEC).

1.5.7 Security Services Portability/Replacability

The CORBA Security specification includes replacability conformance options.

The objects supporting the security mechanism (PrincipalAuthenticator, Vault and Security Context) can be replaced to support the mechanisms in this specification. However, if logon outside the object system is supported, this will need to provide credentials including the security information needed by the CSI mechanism(s) used.

If the invocation access policy is replaced, this can utilise privileges transmitted using CSI protocols. However, if an ORB wishes to control access on invocations using local (e.g. operating system) attributes, then mapping of attributes prior to calling the Access Decision object is needed. An extension to the replacability point is proposed to cover this optional facility.

1.5.8 Performance

Security should not impose an unacceptable performance overhead, particularly for normal commercial levels of security, although a greater performance overhead may occur as higher levels of security are implemented.

Details of the performance overhead depend on the mechanism used and its implementation. However, in this specification:

- sufficient information can be carried in the IOR so that the client knows what security the target supports so does not have to negotiate protocols and options with it.
- the mechanisms used in this specification allow the initial_context_token to be transmitted with first message if mutual authentication is not needed.

1.5.9 Assurance

A security implementation may need to meet Evaluation criteria for assurance. The CORBA Security specification specifies guidelines for a trustworthy system. The choice of security mechanism and algorithms affects the way the CORBA security system can withstand attacks.

1.5.10 Identifying Encumbered Technology

This specification includes technology which is encumbered to some extent.

Use of public key technology helps large scale, particularly inter-enterprise interoperability.

- manage the distribution of cryptographic keys across large networks securely and without undue administrative overheads.

1.5.5 Flexibility of Security Policy

The security policies required varies from enterprise to enterprise, so choices should be allowed, though standard policies should be supported for common secure interoperability.

Access Policies

At CSI levels 0 and 1, the access_id is the only privilege attribute supported. The standard DomainAccessPolicy defined in CORBASEC (or other access policies) can be used with only this privilege.

At CSI level 2, conformant ORBs are able to transmit further privilege attributes (such as role and group - see chapter 2), so the DomainAccessPolicy (and other access policies) can be used with these privileges also.

The protocol at level 2 is designed to allow transmission of further privileges, including user defined ones and security clearances as needed for multi-level secure systems. If received by a conformant ORB, they will be available for access control at the target. However, conformant ORBs need not transmit them, so use of such privileges is subject to the agreement between the systems.

The mechanisms defined in this interoperability standard also allow a wider range of privileges etc to be supported and therefore other access policies to be used. However, interoperability with all other conformant ORBs is not guaranteed in this case.

Audit Policies

All CSI levels provide an audit_id which can be used in audit policies.

CSI level 2 can transmit an audit_id which is anonymous to all but audit administrators.

1.5.6 Application Portability

Application portability is an important OMG requirement. The many applications which are unaware of security will continue to be portable.

Applications which enforce their own security policies should still be portable across ORBs supporting common secure interoperability if the access and audit policies they use rely only on security attributes which are mandatory in the chosen CSI level.

- support of consistent policies for which principals should be able to access what sort of information within a security domain that includes heterogeneous systems.

  For this specification, it requires the ability to transmit consistent privilege and other attributes between ORBs to support these policies. Level 0 and 1 conformant ORBs can transmit identities, level 2 conformant ORBs can transmit a range of privilege attributes.

  These can be the ones used in existing systems, though system specific ones will not be usable in other systems

- fit with existing logons (so extra logons are not needed) and with existing user databases (to reduce the user administration burden).

  Log on needs to result in credentials which include the information required to support the specified security mechanisms. Note that single logon with secure messaging, web etc generally requires use of public key based mechanisms. Also, if non-repudiation is supported, they will also need to include the security information required to support the non-repudiation mechanism - normally a public key one.

  Also, interoperating with non-object systems may require, for example, a CORBA object implementation which calls a non-CORBA application to be able to delegate incoming credentials (assuming compatible security mechanisms.)

Fit with all non-object systems is clearly not possible if such a system uses security mechanisms which are incompatible with the one used in the object system. Such systems may be able to use CORBA Security, but will not be able to interoperate using the common secure interoperability standard.

This specification includes an interoperability level which supports privileges and also a public key (as well as a secret key) mechanism to support these requirements.

1.5.4 Scalability

It should be possible to provide security for a range of systems from small, local systems to large intra- and inter-enterprise ones. As in CORBASEC, for larger systems, it should be possible to:

- base access controls on the privilege attributes of users such as roles or groups (rather than individual identities) to reduce administrative costs.

  This specification includes the transmission of such privilege attributes in CSI level 2.

- have a number of security domains which enforce different security policy details, but support interworking between them subject to policy. (The CORBA Security specification includes the architecture for such inter-domain working, though neither it, nor this specification define interface for this.)

1.5.2 International Deployment

International deployment requires that the security mechanisms and algorithms chosen can be used worldwide in countries which are subject to different national regulatory controls on the use of cryptography. It also requires that they can also be used across international boundaries. International deployment may also be affected by export control regulations and other issues.

Requirements distilled from the key regulations affecting international deployment include:

- keep the amount of information which must be encrypted for confidentiality to a minimum. In general, encryption of keys is acceptable, but encryption of other data may not be.

  For this reason, encryption of security attributes is undesirable. At CSI level 2, where more attributes are generally needed, the CSI-ECMA protocol therefore separates the part of the security tokens concerned with key distribution from the part used to carry privileges, so the latter part does not need to be encrypted.

- be able to use identities for auditing which are anonymous, except to the auditor.

  For this reason, identities used for access control and audit may need to be different. A separate audit_id can be transmitted at level 2.

- allow use of different cryptographic algorithms, with different lengths of keys for specified functions to meet export and use regulations in different countries.

  The specification defines cryptographic profiles which allow for different cases. The mandatory one provides data integrity only, as this is generally easier to deploy internationally.

Note that there may be further requirements on secure ORB products to ensure that they are exportable. For example, they must not allow easy/uncontrolled replacement of cryptographic algorithms. This affects the construction of the system, but not this interoperability standard, so is not considered further here.

Other restrictions on the use of algorithms and security mechanisms are highlighted in *Identifying Encumbered Technology* (1.5.10) below. For example, the DES algorithm is subject to export controls, while RSA requires licensing in some countries. The MIT version of the Kerberos technology widely used in the USA is also subject to export controls.

1.5.3 Consistency

It should be possible to provide consistent security across the distributed object system and also with associated legacy and other non-object systems. This includes:

- standardise on strong confidentiality and integrity, which customers want, but will be subject to export controls in most countries and to deployment regulations in some. Leave vendors and customers to sort out the problems.

This specification makes only the first of these options mandatory, though implementors of all profiles may choose to support other profiles also.

1.4 Conformance to External Security Mechanisms

This specification uses definitions of protocols defined in other standards documents. This specification refers to particular versions of these standards, as this is needed for interoperability. If the versions of these external documents change in future, there may be a need to update this specification so that it is in line with the then most accepted external version of these standards.

1.5 Response to Requirements

The Request for Proposals on Common Secure IIOP specifies requirements for standard security mechanisms, simple delegation and international deployment. It also requires submittors to identify encumbered technology and any changes required to the CORBA Security SECIOP protocol. As a particularisation of the CORBA Security specification, this specification also is subject to the relevant requirements of that specification, which came from OSTF RFP3.

This section lists the key requirements for common secure interoperability from both these sources and how this specification responds to these requirements.

1.5.1 CORBA Standard Security Mechanisms

Standard CORBA security mechanisms are required so that ORBs can interoperate securely at all.

This specification includes three protocols to meet different circumstances as described above. One is mandatory and all conformant ORBs must support it. Interoperability between conformant ORBs is always possible using this, though the facilities supported when using it are limited.

Interoperability also requires common use of cryptographic algorithms. A number of cryptographic profiles are specified to meet the needs of different markets and countries. One is mandatory and interoperability between conformant ORBs is always possible using this, though it provides data integrity, but not confidentiality.

Where multiple mechanisms and cryptographic profiles are supported by both ORBs, the client and target object must agree which to use. In this specification, this is done by the client looking at the security mechanism tag in the target object reference and choosing an appropriate mechanism and profile which both support. (In future, negotiation of mechanisms may be supported.)

Acceptability

Kerberos has been available in the market place longest and so is most acceptable from the view of wide deployment, so known technology. However, it is secret key only and does not support significant facilities in CORBASEC such as carrying privileges (other than access_id) for access control. Also, the lack of restrictions on delegation causes trust problems in large systems.

The public key technology scene is younger and more volatile. No protocol has yet achieved an obvious dominance of the whole market. SSL is strong in web markets, though does not support facilities (e.g. privileges) which are required in CORBA. SPKM and SESAME based products are being used commercially, and are growing in the market, though do not yet have the number of years of proven use that Kerberos has.

Conclusion

GSS Kerberos is specified as the mandatory protocol for common secure interoperability as Kerberos is widely available and most vendors can support it. However, it does not provide all facilities required and is secret key only.

CSI-ECMA is specified as the protocol to provide support for the full set of CORBASEC security facilities using public key or secret key technology. The ECMA protocol is currently the only standard protocol which meets these requirements, and it is deployed in SESAME based products.

SPKM is specified as a simpler public key protocol suitable for applications where access and audit policies are fairly static and at each stage in a chain of object invocations, the policies depend only on the identity of the immediate invoker, not the initiator of the chain.

1.3.2 Cryptographic Profiles

Currently, different cryptographic algorithms, and/or different key lengths are required to meet export controls and regulations on use of cryptography in various countries - see 1.5.2. Some vendors produce more than one version of secure products for different markets, though are increasingly reluctant to do this. For common secure interoperability, a particular cryptographic profile is needed. Some options are:

- standardise integrity only for user data, not confidentiality. If done using MD5, say, this is likely to be exportable and generally deployable, but doesn't provide confidentiality when interoperating. So this does not provide the functionality which some users will want.
- standardise integrity and confidentiality using weak keys only. This provides the required functionality, in a way which can generally be exported, but does not provide the strength of protection needed by some customers. Also, products using it may be subject to import controls or other regulations in some countries

1.3.1 Choice of Protocol and key technology

The choice of protocol to use depends on:

- the facilities required. For example, is delegation needed? Are access policies based on privileges such as roles needed?
- the type of technology wanted for key distribution. This is likely to depend on other functionality to be supported with consistent management of keys, scalability to inter-enterprise working etc
- the general acceptability in the market place of the particular protocols proposed

Functionality

The CORBA Security specification defines functionality for secure CORBA compliant systems. This includes the use of a principal's attributes for access and audit policies and delegation of these attributes through a chain of encapsulated objects.

Since adoption of the CORBA Security specification, both vendors and users have expressed their intent to provide/use privilege based, e.g. role based, access control. (However, some will be content to use identity based controls only.) Also, the design of many, but not all, object applications requires some delegation of attributes.

These features should therefore be included in this common secure interoperability specification to support the full CORBA Security specification in environments with different ORBs. (Privileges and restricted delegation are currently available for non-object systems in DCE and SESAME based products.)

Key Distribution Technology

There is a current strong move in the market place towards public key technology because of its use in mail and web environments where inter-enterprise working is more common. Also, enterprises want common management of user information, including keys, and this is difficult if secret key technology is used for object invocations and public key for non-repudiation, mail etc. Even within an object environment, the CORBASEC specification includes both non-repudiation and secure object invocations so commonality of key management would normally require use of public keys.

The particular public key protocols being specified for web etc use do not provide significant CORBA security functionality including privileges and delegation. There is no sign that they will do so soon. (When discussed at the OMG meeting Washington in June 1996, estimates of 18 months to 2 years were suggested).

In the client-server market both SPKM and SESAME based commercial products support public key protocols for secure session oriented applications.

Not all users want public key protocols, so a secret key based protocol is also needed.

1.2.4 Cryptographic Profiles

Security mechanisms use cryptography in the establishment of a secure association between a client and target and in protecting the data between them. Different cryptographic algorithms are used to support particular security functions depending on the type of mechanism used and also the regulations on use of cryptography - see issue in 1.3.2. The combination of algorithms used to provide particular security using a particular mechanism is called a cryptographic profile.

CSI only mandates profiles which provide integrity, but not confidentiality of user data

1.2.5 Conformance

To claim conformance to this specification, CSI level 1 functionality must be provided using the GSS Kerberos protocol with the MD5 cryptographic profile.

The following additional conformance can be claimed:

- CSI-ECMA Public Key at level 0, 1 or 2 by providing the specified level of CSI functionality using the CSI-ECMA protocol with the public key option (mechanism CSI_ECMA_Public).
- CSI-ECMA Secret Key at level 0, 1 or 2 by providing the specified CSI level using the CSI-ECMA protocol with the secret key option (mechanism CSI_ECMA_Secret).
- CSI-ECMA Hybrid at level 0, 1 or 2 by providing the specified CSI level using the CSI-ECMA protocol with the hybrid key option (mechanism CSI-ECMA-Hybrid).
- SPKM at level 0 by providing the specified CSI levelusing the SPKM protocol (mechanism SPKM_1 and optionally also SPKM_2)

In addition, a conformant ORB must specify all the cryptographic profiles it supports.

The following table shows which CSI functionality is supported with which protocols.

Protocol / CSI Level	SPKM	GSSKerberos	CSI-ECMA
0	Supported	Supported	Supported
1	Not supported	Supported (Mandatory)	Supported
2	Not supported	Not supported	Supported

1.3 Issues

There are two issues which the submittors wish to be particularly visible to OMG members as they have been subject of debate both inside and outside OMG. In both cases, a particular resolution of the issue is specified, which the group believe best meets OMG's current needs given the other constraints.

All types of key distribution can be used to support all the facilities in CORBA Security for secure object invocations (though public key is almost universally used for non-repudiation). So the choice of mechanism to use depends on a customers requirements, for example for fit with other systems and for scalability to inter-enterprise working (where sharing secret keys between enterprises is likely to be deprecated). Issues on the choice of technology are explained further in 1.3.

1.2.3 Common Security Protocols

These define the details of the tokens in the SECIOP messages. Three protocols are defined:

SPKM Protocol

This protocol supports identity based policies without delegation (CSI level 0) using public key technology for keys assigned to both principals and trusted authorities.

The SPKM protocol is based on the definition in [SPKMMECH].

GSS Kerberos Protocol

This protocol supports identity based policies with unrestricted delegation (CSI level 1) using secret key technology for keys assined to both prinicipals and trusted authorities. It is possible to use it without delegation (so providing CSI level 0).

The GSS Kerberos protocol is based on the [GSSKRB5] which itself is a profile of [KERBV5].

CSI-ECMA protocol

This protocol supports identity and privilege based policies with controlled delegation (CSI level 2). It can be used with identity, but no other privileges and without delegation restrictions if the administrator permits this (CSI level 1) and can be used without delegation (CSI level 0).

For keys assigned to principals, it has two options - it can use either secret or public key technology. It uses public key technology for keys assigned to trusted authorities.

The CSI-ECMA protocol is based on the ECMA GSS-API Mechanism as defined in ECMA 235, but is a significant subset of this - the SESAME profile as defined in [SESAMEMECH]. It is designed to allow addition of new mechanism options in future; some of these are already defined in ECMA 235.

Choice of Protocol

The choice of protocol to use depends on the mechanism type required (see 1.2.2) and the facilities required by the range of applications expected to use it.

As delegation is not restricted, once an initiator has delegated his identity, it must trust the objects it calls not to abuse its delegated rights to act as the initiator. In practice, this will limit the type of environment in which level 1 should be used to relatively closed environments.

An example of an application environments which can use level 1 facilities is a back office system protected by firewalls where identity based policies are acceptable.

Identity & privilege based policies with controlled delegation (CSI level 2)

At this level, attributes of initiating principals passed from client to target can include separate access and audit identities and a range of privileges such as roles and groups. Delegation of these attributes to other objects is possible, but is subject to restrictions, so the initiating principal can control their use. Optionally, composite delegation is supported, so the attributes of more than one principal can be transmitted. It therefore provides interoperability for ORBs conforming to all CORBA Security functionality.

Access and audit policies are based on the attributes of initiating principals. At this level, a wider range of policies can be supported e.g. role based access controls, mandatory access controls using the initiating principal's security clearance.

At this level, an initiator needs to trust those targets which it has allowed to use its attributes not to abuse these, but it does not have to trust these targets not to delegate the attributes outside the trusted set of targets, as the delegation controls can be used to prevent this.

This level can be used for a wide range of applications in large enterprise and inter-enterprise networks.

1.2.2 Key Distribution Types

Security mechanisms use cryptography in the establishment of a secure association between a client and target and in protecting the data between them. Security mechanism differ in the type of cryptography they use, particularly for distribution of keys. (Keys are assigned to clients and targets and also to trusted authorities). Three types of key distribution are defined in this specification:

- **Secret key** ones which use secret key technology for distribution of keys for principals. Where CSI mechanisms use this, it is based on Kerberos V5 as defined in [KERBV5].
- **Public key** ones which use public key technology for distribution of keys for principals, though may use secret key technology for message protection. Where CSI mechanisms use this, it is based on the ECMA and SPKM definitions in [ECMAMECH] and [SPKMMECH] which have common profile for the key distribution part of the protocol when using public key technology only.
- **Hybrid** ones which use secret key technology for key distribution for principals within an administration domain, and public key technology for key distribution for trusted authorities, and hence between domains.

1.2 Common Secure Interoperability Description

1.2.1 Common secure interoperability levels

The Common Secure Interoperability specification defines three functionality levels and outlines the type of applications and environments where these are recommended. An example of the difference in use of the three levels is explained in chapter 2.

All levels can be used in distributed secure CORBA compliant object systems where clients and objects may run on different ORBs and different operating systems. At all levels, security functionality supported during an object request includes (mutual) authentication between client and target and protection of messages - for integrity, and when using an appropriate cryptographic profile, also for confidentiality.

An ORB conforming to CSI level 2 can support all the security functionality described in the CORBA Security specification. Facilities are more restricted at levels 0 and 1. The three levels are:

Identity based policies without delegation (CSI level 0)

At this level, only the identity (no other attributes) of the initiating principal is transmitted from the client to the target, and this cannot be delegated to further objects. If further objects are called, the identity will be that of the intermediate object, not the initiator of the chain of object calls.

Access and audit policies at this level are based on the identity of the immediate invoker. So access and audit policies in encapsulated objects which depend on the initiator of the chain, can only be used at the point of entry to the object system, not in further objects encapsulated by it.

As the attributes of principals are not delegated, environments do not need to be trusted not to pass on principal information which should be controlled.

Examples of applications which can use level 0 facilities are wrapped legacy applications and telephone switches. If a CSI level 0 ORB also supports non-repudiation, it can also be used for other types of applications such as electronic funds transfer.

Identity based policies with unrestricted delegation (CSI level 1)

At this level, only the identity (no other attributes) of the initiating principal is transmitted from the client to the target. The identity can be delegated to other objects on further object invocations, and there are no restrictions on its delegation, so intermediate objects can impersonate the user. (This is the impersonation form of simple delegation defined in CORBASEC.)

Access and audit policies at this level can be based on the identity of the initiating principal or immediate invoker, depending on the delegation policy.

Introduction 1

1.1 Scope

The CORBA Security specification [CORBASEC] describes the model and architecture for security in CORBA compliant systems and specifies the IDL interfaces and security functionality levels and options. It allows support of a range of security policies and mechanisms.

It also includes a specification of a secure inter-ORB protocol (SECIOP) for use with the CORBA 2 GIOP/IIOP interoperability protocol and security tags for the Interoperable Object Reference (IOR). The CORBA Security specification allows use of different security mechanisms and policies.

This Common Secure Interoperability specification defines the standards for common secure interoperability when using GIOP/IIOP by defining:

- standard security mechanisms and associated cryptographic algorithms.
- details of the SECIOP protocol messages and IOR security tags when using these mechanisms and algorithms.
- the security functionality supported when interoperating using these security mechanisms.

It also defines what is required to conform to the mandatory and optional parts of the specification. Different conformance points provide different functionality and may use different mechanisms for secure interoperability.

Note: this CSI specification is confined to secure interoperability of object requests and replies via the GIOP/IIOP protocol. It does not cover interoperability of particular types of data, for example, non-repudiation tokens.

Table of Contents

Trademarks
All trademarks acknowledged.

Contacts

Dan Frantz
Digital Equipment Corporation
110 Spit Brook Rd, 2K02-2/R80
Nashua, New Hampshire 03062-2698
USA

E-mail: frantz@send.enet.dec.com

Anne Hopkins
Hewlett-Packard Company
Chelmsford System Software Lab
300 Apollo Drive CHR-03-DC
Chelmsford MA 01824

E-mail: ahop@apollo.hp.com

Belinda Fairthorne
ICL
Lovelace Road
Bracknell
Berkshire
United Kingdom

E-mail: belinda@iclnet.co.uk

Guangxing Li
Nortel Technology
London Road
Harlow CM17
United Kingdom

E-mail: G.Li@nortel.co.uk

Christian Ammon
Siemens Nixdorf
Otto-Hahn-Ring 6
81730 Munich
Germany

E-mail: Christian.Ammon@mch.sni.de

Kent Salmond
Tandem Computers Incorporated
10501 North Tantau
Cupertino, CA 95014
USA

E-mail: salmond_kent @loc201.Tandem.COM

Matt Stillerman
Odyssey Research Associates, Inc.
301 Dates Drive
Ithaca NY 14850
USA

E-mail: matt@oracorp.com

Hervé Lejeune
Groupe Bull - B1/121
1, rue de Provence
38432 Echirolles Cedex
France

E-mail: H.Lejeune@frec.bull.fr

Bob Blakley
International Business Machines Corporation
11400 Burnet Road
Mail Stop 9356
Austin TX 78758

E-mail: blakley@vnet.ibm.com

Annrai O'Toole
Iona Technologies
The Iona Building
8-10 Lower Pembroke Street
Dublin 2
Ireland

E-mail: aotoole@iona.ie

Rick Wessman
Oracle Corporation
500 Oracle Parkway
Box 659410
Redwood Shores, CA 94065

E-mail: rwessman@us.oracle.com

Rogit Garg
Sun Microsystems, Inc.
2550 Garcia Ave, MS UMPK18-209
Mountain View, CA 94043-1100
USA

E-mail: rohit.garg@eng.sun.com

Ted Ralston
Black Watch Technology Incorporated
2-212 CASE Center - Syracuse University
Syracuse, NY 13244-4100
USA

E-mail: ted@blackwatch.com

Mark Wales
National Security Agency
9800 Savage Road MS R23
Fort Neade
Maryland 20755-6000

E-mail: mgw@epoch.ncsc.mil

Common Secure Interoperability

Digital Equipment Corporation
Groupe Bull
Hewlett-Packard Company
International Business Machines Corporation
International Computers Limited
Iona
Nortel Ltd
Oracle
Siemens Nixdorf Informationssysteme AG
Sunsoft, Inc
Tandem Computer Incorporated

In collaboration with
Black Watch
National Security Agency
Odyssey Research Associates Inc.

July 1996

OMG Document Number: orbos/96-06-20