
M. K. F. Lai

APPROVED FOR PUBLIC RELEASE

© Commonwealth of Australia

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

M.K.F. Lai

Information Technology Division
Electronics and Surveillance Research Laboratory

DSTO-RR-0063

ABSTRACT (U)
This paper is part of the documentation series produced under the HQADF sponsored task “D6: A Security Architecture for Large, Distributed Multimedia Systems”. It proposes to use the Message Security Protocol (MSP) for providing the security service support to the Directory Access Protocol (DAP). This may be viewed as an interim option for the directory service implementation before the militarised DAP is finalised and its trusted directory user agent becomes available. Specifically, this paper focuses on the necessary protocol elements of MSP for supporting the security requirements of the DAP.

APPROVED FOR PUBLIC RELEASE
UNCLASSIFIED

Published by

DSTO Electronics and Surveillance Research Laboratory
PO Box 1500
Salisbury, South Australia, 5108

Telephone: (08) 259 7053
Fax:(08) 259 5619

© Commonwealth of Australia 1996
AR No. 009-444
January 1996

APPROVED FOR PUBLIC RELEASE

UNCLASSIFIED

Executive Summary (U)

This paper is part of the document series produced under the HQADF sponsored task “D6: A Security Architecture for Large, Distributed Multimedia Systems”. The earlier DSTO paper [1] has discussed the rationale for creating a national specific extension field within the Message Security Protocol (MSP) [2] and its intended usage. This paper is a subsequent exploration of the current MSP technology for supporting one of the applications identified in [1].

A message handling system is a communications infrastructure that provides users the ability to exchange and distribute messages electronically. In order to provide messaging users the necessary flexibility and efficiency, the users need to access the directory service. An analogue is the accessing of the ‘white pages’ by the telephone users. While the security of messaging systems is being addressed in terms of the ‘writer-to-reader’ security, very little has been proposed or developed for secure directory access. This paper aims to discuss the directory access security based on the developing or adopted military security technologies for messaging. Specifically, this paper addresses the Directory Access Protocol (DAP) [7] between a user and the directory. This protocol allows the user to request the directory operations (including reading, searching, and modifying the directory information). It also provides the directory the procedure to respond to the user’s request. The MSP [2] has been designed to address the security requirements for electronic message writers and readers. These requirements are exactly some of those that also are required by the directory operation requests and responses of the DAP.

While the Combined Communications Electronics Board (CCEB) nations have approved the MSP adoption and the MSP user agent implementation is starting to appear in the market place, the ACP133 (which is the military standard that provides the directory security service definition) remains to be finalised. The security services embedded in the current DAP are not as strong as those provided by the MSP. There are additional facilities defined in the MSP which are also useful for the directory services. It is therefore reasonable to infer that the CCEB would wait for the completion of ACP133 before it decides on the DAP adoption.

In the meantime, many X.500 compliant directory systems are appearing to meet the user demand, although the security services (provided by the associated directory user and system agents) are not necessarily trusted in terms of the military requirement. To meet the immediate military requirement for querying X.500 directory systems of various
classification levels and modifying their entries, this paper aims to outline an interim option to use the MSP for providing the following security service support to the DAP:

- user identity authentication - associated with the directory binding request;
- directory identity authentication - associated with the directory binding response;
- non-repudiation with proof of operation request - associated with all the read, search, and modify operation requests;
- non-repudiation with proof of read or search result origin - associated with all the read and search operation results;
- non-repudiation with proof of delivery of read or search result - associated with all the read and search operation results;
- confidentiality - associated with all the read, search, and modify operation requests and all the read and search operation results.

The DAP (which specifies the directory operation argument and result syntax) is used by the Directory User Agent (DUA) to request an operation, and by the Directory System Agent (DSA) to respond to the request. This paper suggests that

- a MSP user agent (MSPUA) (for the directory user) resides between the DUA and the OSI application services; and
- a MSP directory agent (MSPDA) (for the directory) resides between the DSA and the OSI application services.

To provide the above security services, the MSPUA and MSPDA use the protocol Msp to communicate between themselves. The MSPUA uses an instance of the Msp to encapsulate a directory argument of the DUA as an extension value within the Extensions field. Upon its arrival, the MSPDA processes the Msp instance and checks that the originator has applied the claimed security services. If the check succeeds, it then passes the directory argument (found in the encapsulation) to the DSA. When the DSA responds to the directory argument, the MSPDA similarly encapsulates the responding directory result of the DSA in an instance of the Msp which then is sent to the MSPUA.

For each of the directory operation arguments and results of the DAP, it is required that an instance of the Msp is defined for its encapsulation. This paper presents the detail of these Msp instances in the appendix (Section 8). It also explains that the encapsulation occurs at the Extensions field of the Msp. The Extensions field is more flexible to use and adopt because it is required only that a new Extension is registered (so that an extnID is assigned), along with its syntax and the elements of procedure for the originator and recipient [2]. Finally, this paper discusses some protected DAP associations (that become possible because of the security protection provided by the MSP), as well as the current limitations. The option of using the MSP therefore may be viewed as an interim step in the migration towards the fine grain directory entry access control security services.


Lai M.K.F.
Information Technology Division

Dr. Lai received his B.Sc. (Mathematical Science) First Class Honours degree in 1984, followed by his Ph.D. in Combinatorial Group Theory completed in 1987, both from the University of London. He is currently a Senior Research Scientist in the Information Security Section of Trusted Computer Systems Group at DSTO.
This is a blank page.
Contents

1 INTRODUCTION  1

2 X.500 DIRECTORY AND THE DIRECTORY USER OPERATIONS  3
  2.1 Directory Access Protocol (DAP)  3
  2.2 Application Services Used by the DAP  4

3 USING MSP TO PROVIDE SECURITY SUPPORT TO DAP  7
  3.1 The Basic MSP Approach  7
  3.2 Positioning MSP in the Protocol Stack  8
  3.3 The Composition of MSP  10
  3.4 Using the Extensions to Support Directory Operations  10

4 MSP-SUPPORTED DIRECTORY BIND OPERATIONS  13
  4.1 MSP-Directory Bind Argument  13
     4.1.1 Originator Security Data  14
     4.1.2 Signature Block  14
     4.1.3 Per Recipient Token  15
     4.1.4 Directory Bind Argument Extension  15
  4.2 MSP-Directory Bind Result  16
  4.3 MSP-Directory Bind Errors  17

5 SOME MSP-PROTECTED DUA-DSA ASSOCIATIONS  19
  5.1 DUA-DSA System High  19
  5.2 Remote DUA-DSA System High  19
  5.3 High DUA and Low DSA  19
  5.4 Classified DUA and Unclassified DSA  20
  5.5 Illegal Associations  21

6 CONCLUSION  23

7 REFERENCES  25

8 APPENDIX: THE MSP-SUPPORTED DIRECTORY OPERATIONS  27
  8.1 Binding Operations  27
     8.1.1 MSP-Directory Bind Argument  27
     8.1.2 MSP-Directory Bind Result  28
     8.1.3 MSP-Directory Bind Errors  29
  8.2 Directory Read Operations  29
     8.2.1 MSP-Read Argument  30
     8.2.2 MSP-Read Result  30
     8.2.3 MSP-Errors  31
  8.3 Other Directory Operations  32
     8.3.1 Compare Operations  32
        8.3.1.1 Compare Argument  32

UNCLASSIFIED
1 Introduction

This paper is part of the document series produced under the HQADF\(^1\) sponsored task “D6: A Security Architecture for Large, Distributed Multimedia Systems”. The earlier DSTO paper [1] has discussed the rationale for creating a national specific extension field within the Message Security Protocol (MSP) [2] and its intended usage. This paper is a subsequent exploration of the current MSP technology for supporting one of the applications identified in [1]. Specifically, this paper addresses the directory services of secure querying and updating directory information.

A directory consists of a set of systems which provide users the service associated with the information that they collectively hold. This service is the simple capability to retrieve and modify information stored in the directory. The directory service, protocols for communications between systems of the directory, and the abstract models of the directory information are specified in the X.500 series of standards (described in [3]). For the purpose of this paper, a directory loosely is called a X.500 directory if it can provide the X.500 service, can support the X.500 protocols, and can handle information based on the DIB models.

This paper specifically addresses the protocol (called the Directory Access Protocol (DAP)) between a user and the directory. This protocol allows the user to request the directory operations (including reading, searching, and modifying the directory information). It also provides the directory the procedure to respond to the user's request.

The Message Security Protocol (MSP) [2] has been designed to address the security requirements for electronic message writers and readers. It focuses on the confidentiality, data-origin authentication, and non-repudiation with proof of origin and delivery security services for a message between the writer and reader. These are exactly some of the security services that also are required by the directory operation requests and responses of the DAP.

While the Combined Communications Electronics Board (CCEB) nations have approved the MSP adoption and the MSP user agent implementation is starting to appear in the marketplace, the ACP133 (which is the military standard that provides the directory security service definition) remains to be finalised. The security services embedded in the current DAP are not as strong as those provided by the MSP. Most notably, the confidentiality service has not been defined for the DAP. There are additional facilities defined in the MSP which are also useful for the directory services. These include partition or local rule based access control,

\(^1\) Headquarters Australian Defence Force.
and security labelling (which can be used for addressing the users’ privilege requirements). It is therefore reasonable to infer that the CCEB would wait for the completion of ACP133 before it decides on the DAP adoption.

In the meantime, many X.500 compliant directory systems are appearing to meet the user demand, although the security services (provided by the associated directory user and system agents) are not necessarily trusted in terms of the military requirement. This user demand is driven by the current introduction of the Defense Message System (DMS) in the US, the Defence Message and Directory System (DMDS) in Australia, and other similar programs in the allied nations [4]. To meet the immediate military requirement for querying X.500 directory systems of various classification levels and modifying their entries, this paper aims to outline an interim option to use the MSP for providing the following security service support to the DAP:

- user identity authentication - associated with the directory binding request;
- directory identity authentication - associated with the directory binding response;
- non-repudiation with proof of operation request - associated with all the read, search, and modify operation requests;
- non-repudiation with proof of read or search result origin - associated with all the read and search operation results;
- non-repudiation with proof of delivery of read or search result - associated with all the read and search operation results;
- confidentiality - associated with all the read, search, and modify operation requests and all the read and search operation results.

Furthermore, the partition rule based access control (PRBAC) and local rule based access control (LRBAC) facilities (defined in [2]) of MSP also may be associated with a specific user’s directory operation requests (including bind, read, search, and modify requests) to a given directory system. In this way, limited control for accessing the directory system can be realised.

In the discussion of the above security services in the following sections, this paper assumes the existence of the trusted MSP user agent and the trusted path between the agent and the user. They are prerequisites for implementing the primary MSP application (namely the writer-to-reader secure messaging service) and so are not specific to the additional MSP application for the directory service (which is the focus of this paper). The composition of trusted components for providing the trusted MSP user agent is outside the scope of this paper. The issues related to the trusted composition will be discussed elsewhere [5]. This paper focuses only on the protocol elements of MSP necessary for supporting the security requirements of the DAP.
2 X.500 Directory and the Directory User Operations

A directory consists of a set of systems which provide users the service associated with the information that they collectively hold. This service is the simple capability to retrieve and modify information stored in the directory. The collective information is known as the directory information base (DIB) of the directory [3]. The directory service, protocols for communications between systems of the directory, and the abstract models of the DIB are specified in the X.500 series of standards (described in [3]). For the purpose of this paper, a directory loosely is called an X.500 directory if it can provide the X.500 service, can support the X.500 protocols, and can handle information based on the DIB models.

The directory service is provided to a user by means of a number of directory operations [6]. These directory operations are of four different kinds, all of which are initiated by the user:

- directory bind operations, which provide the binding and subsequent unbinding for an association between the requesting user and the directory;
- directory read operations, which interrogate a single directory entry during the association;
- directory search operations, which interrogate potentially several directory entries during the association; and
- directory modify operations during the association.

The directory bind operations include

- DirectoryBindArgument and DirectoryUnbind (initiated from the user); and
- DirectoryBindResult or DirectoryBindError (used by the directory to respond).

The directory read operations include

- ReadArgument and CompareArgument (initiated from the user); and
- ReadResult, CompareResult, or ERRORS (used by the directory to respond).

The directory search operations include

- SearchArgument and ListArgument (initiated from the user); and
- SearchResult, ListResult or ERRORS (used by the directory to respond).

The directory modify operations include

- AddEntryArgument, RemoveEntryArgument, ModifyEntryArgument, and ModifyDNArgument (initiated from the user); and
- AddEntryResult, RemoveEntryResult, ModifyEntryResult, ModifyDNResult, or ERRORS (used by the directory to respond).

2.1 Directory Access Protocol (DAP)

The requests and responses of the above directory operations are carried out by the directory protocols. In the case of the user-directory association, which is the focus of this paper, the responsible directory protocol that carries out the user's operation requests (in terms of
operation arguments) and the directory's responses (in terms of operation results or **ERRORS**) is the directory access protocol (DAP). The DAP is defined in the standard [7].

In an actual implementation, the DAP is used by the two entities called the directory user agent (DUA) and directory system agent (DSA) instead of the user-directory pair. There is precisely one DUA for each user. The DUA performs the directory operation requests on behalf of the user. It also displays the subsequent operation responses (from the directory) to the user. On the other hand, the DSA could be part or the whole of the directory, depending on how much of the DIB it contains. Some DSAs are built from relational databases, others from hierarchical databases, and others from purpose built databases. The only common (DSA) characteristic (that all these many different types of database must share) is the ability to respond to the directory operation requests, and to understand the directory protocols, including specifically the DAP. The following diagram (Figure 1) represents the DAP interaction between a DUA and a DSA [8].

![Diagram of DAP Interaction]

**Figure 1  DAP Interaction [8]**

### 2.2 Application Services Used by the DAP

In realising the DAP interaction between a DUA and a DSA across the network, the X.500 directory specifications [7] assumes that the DUA and the DSA use two OSI application services. They are the association control service (ACSE) [9] and the remote operations service (ROSE) [10].

The ACSE is used to establish the association between the DUA and the DSA. The directory bind operations are mapped onto the parameters of ACSE. The DUA is the initiator of the directory binding operation (**directoryBindArgument**) and therefore of the association.

The ROSE provides a simple request-response type of service. The DUA provides one of the directory read, search or modify operation requests (such as **ReadArgument**,
SearchArgument, or AddEntryArgument) to the ROSE in terms of the parameters of the ROSE service primitives. The ROSE packages the request correctly, sends it to the peer ROSE of the DSA, and gives its DUA the reply when it arrives.

The use of the ACSE and ROSE services is summarised in the following figure (Figure 2).

![Diagram](image)

Figure 2  DAP Using OSI Application Services
This is a blank page.
3 Using MSP to Provide Security Support to DAP

This paper intends to outline an option to use the Message Security Protocol (MSP) [2] for providing the following security service support to the DAP:

- user identity authentication - associated with DirectoryBindArgument;
- directory identity authentication - associated with DirectoryBindResult;
- non-repudiation with proof of operation request - associated with all the read, search, and modify operation arguments;
- non-repudiation with proof of read or search result origin - associated with all the read and search operation results;
- non-repudiation with proof of delivery of read or search result - associated with all the read and search operation results;
- confidentiality - associated with all the read, search, and modify operation arguments and all the read and search operation results.²

For the authentication and non-repudiation services, the signature mechanism (in terms of signature generation) of MSP is applied to the directory operation arguments and results. For confidentiality, the confidentiality mechanism (in terms of the exchange of encryption key) of MSP is provided to protect the read, search, and modify operation arguments, and the read and search operation results.

Furthermore, the partition rule based access control (PRBAC) and local rule based access control (LRBAC) facilities (defined in [2]) of MSP may be associated with a specific user’s directory operation requests (including bind, read, search, and modify requests) to a given directory system. In this way, limited control for accessing the directory system can be realised. For example, a user may have only the privilege to initiate a read or search operation but not to initiate any modify operation request to the directory. The period of time that a user is allowed to establish a bind operation with the directory system also can be limited through the PRBAC and LRBAC facilities of MSP. These kinds of privileges could be specified easily in the user attribute certificates (which have been defined recently in version 4.0 of MSP).

3.1 The Basic MSP Approach

The MSP provides the security services (including confidentiality, data origin authentication, non-repudiation with proof of origin or delivery, PRBAC, and LRBAC) for a message by

1. encapsulating the message in the protocol Msp; then
2. performing security processing on the message; and then
3. adding a security header containing information associated with the message security processing outputs.

² The modify operation results do not contain any information [6].

The directory operation arguments may be considered as messages from DUA to DSA and the directory operation results also may be considered as messages from DSA to DUA.

For example, the MSP can provide the data origin authentication service for the DirectoryBindArgument and the DirectoryBindResult by the encapsulation, the security processing, and the addition of security header. Therefore, the user identity authentication follows from the authenticated DirectoryBindArgument and the directory identity authentication follows from the authenticated DirectoryBindResult. For the purpose of this paper, denote the MSP-protected DirectoryBindArgument and DirectoryBindResult by MSP-DirectoryBindArgument and MSP-DirectoryBindResult respectively. Similarly, denote the other MSP-protected directory read, search, and modify operations:

ReadArgument, CompareArgument, ListArgument, ReadResult, CompareResult,
ListResult, SearchArgument, SearchResult, ERRORS, AddEntryArgument,
RemoveEntryArgument, ModifyEntryArgument, and ModifyDNAArgument
by

MSP-ReadArgument, MSP-CompareArgument, MSP-ListArgument, MSP-
ReadResult, MSP-CompareResult, MSP-ListResult, MSP-SearchArgument, MSP-
SearchResult, MSP-ERRORS, MSP-AddEntryArgument, MSP-
RemoveEntryArgument, MSP-ModifyEntryArgument, and MSP-ModifyDNAArgument
respectively. The abstract syntax definitions of the above MSP-protected directory operation arguments and results are given in the appendix (Section 8).

3.2 Positioning MSP in the Protocol Stack

The mapping of DAP onto the ACSE and ROSE services is specified in Section 8 of [7]. The DUA should map the DirectoryBindArgument onto the User Information parameter of the A-ASSOCIATE request primitive while the DSA should map the DirectoryBindResult onto the User Information parameter of the A-ASSOCIATE response primitive. Both of the A-ASSOCIATE primitives are provided by the ACSE. Similarly, the directory read, search, and modify operation arguments and results are mapped onto the User Information parameter of the appropriate ROSE service primitives (consisting of RO-INVOLVE, RO-
RESULT, RO-ERROR, RO-REJECT-U and RO-REJECT-P), according to [6] and [10].

Hence, the ACSE service primitives present the obvious point in the protocol stack where the MSP-DirectoryBindArgument and the MSP-DirectoryBindResult can be the substitute for the DirectoryBindArgument and the DirectoryBindResult respectively within the
ASCE User Information parameter. Similarly, the ROSE service primitives also are the point where the MSP-protected directory read, search, and modify operation arguments and results can be the substitute for the corresponding directory read, search, and modify operation arguments and results within the ROSE User Information parameter. In other words, it is suggested that

- a MSP user agent (MSPUA) (for the directory user) resides between the DUA and the OSI application services including the ACSE and ROSE; and
- a 'MSP directory agent (MSPDA)' (for the directory) resides between the DSA and the OSI application services including the ACSE and ROSE.

The MSPUA should be the same as that for a message user who writes or reads a MSP-protected message based on X.400 [11] or ACP123 [12]. The MSPDA would be similar to the MSPUA except that it would not initiate the ACSE or ROSE service primitive requests. Its role is only to respond to the peer MSPUA's operation requests such as the MSP-DirectoryBindArgument or MSP-ReadArgument. For example, it may use the ACSE A-ASSOCIATE response primitive only after the ACSE A-ASSOCIATE indication primitive has been received earlier. The following diagram (Figure 3) shows the placement of the MSPUA and the MSPDA.

![Diagram of the Placement of MSPUA and MSPDA](image_url)

Figure 3  The Placement of MSPUA and MSPDA
3.3 The Composition of MSP

According to the MSP specification [2], the protocol Msp is defined as a sequence of the following optional arguments (fields):

1. originatorSecurityData;
2. signatureBlock;
3. recipientSecurityData;
4. contentDescription;
5. mlControlInformation;
6. extensions;
7. mspSequenceSignatureAlgorithm;
8. mspSequenceSignatureCertificate;
9. encapsulatedContent.

The first eight arguments form the security header of the Msp. The last argument encapsulatedContent (which is an octet string) contains the message being protected. To use the Msp to protect a directory operation argument or result, there are two arguments within the Msp where the directory operation argument or result could reside. One is the encapsulatedContent and the other is the extensions. This paper chooses to use the extensions argument because it would impact the interoperability less than the encapsulatedContent argument does. For its original inclusion in the Msp, it has been argued in [1] and stated in [2] that the extensions argument is intended for use within a local national/domain boundary while remaining global interoperability.

3.4 Using the Extensions to Support Directory Operations

The extensions field is defined in [2] as a sequence of Extension where Extension is specified as follows.

\[
\text{Extension} ::= \text{SEQUENCE} \{ \\
\text{extnID} \text{ OBJECT IDENTIFIER,} \\
\text{critical} \text{ BOOLEAN DEFAULT FALSE,} \\
\text{extnValue} \text{ OCTET STRING } \}
\]

For the extensions, the MSP specification [2] states the extension processing as follows. The MSPUA checks to see if any extensions are present. For each Extension, if the MSPUA recognises the extnID of the Extension, the MSPUA performs the procedures specified as part of the registration of the Extension. If the MSPUA does not recognise the extnID and the critical is FALSE, the MSPUA ignores the Extension. If the MSPUA does not recognise the extnID and the critical is TRUE, the MSPUA terminates processing of the message and a security error has occurred. On the other hand, the MSP specification [2] does not specify
what actions should be taken if the MSPUA does not recognise the `encapsulatedContent`. In other words, any misuse of the `encapsulatedContent` could present a problem related to interoperability.

The `extensions` field is therefore more flexible to use and adopt. To create an `Extension`, it is required only that the `Extension` is registered (so that an `extnID` for the `Extension` can be assigned), along with its syntax and the elements of procedure for the originator and recipient (as stated in [2]). However, to use an `encapsulatedContent`, which is different from those that already exist, a new `encapsulatedContentType` is necessary. This may require the MSP specification to be updated to a new version.

In the rest of this paper, it is assumed that an `Extension` is defined for each of the directory operation arguments and results. For each such `Extension`, the `critical` argument is set to be `TRUE`. This ensures a security error if the receiving MSPDA or MSPUA does not recognise the `Extension`. The `extnValue` is a sequence consisting of the corresponding directory operation argument or result, the current time and the optional protection request. The protection request is considered only when a directory operation argument is initiated. It is not necessary to include the protection request in a directory operation result when responding to an operation request.

For example, let `DirectoryBindArgumentExtn` be the `Extension` defined for the `directoryBindArgument`. It is considered as an instance of the `Extension`. It then may be defined as follows.

```plaintext
DirectoryBindArgumentExtn  Extension  ::=  
          BindArgumentExtnID,  
          TRUE,  
          DirectoryBindArgumentExtnValue } 
DirectoryBindArgumentExtnValue  ::=  SEQUENCE  
                                    { 
                                        directoryBindArgument  DirectoryBindArgument,  
                                        dirDate  UTCTime,  
                                        target  ProtectionRequest OPTIONAL  
                                    } 
ProtectionRequest  ::=  INTEGER  
                      { 
                        none (0),  
                        signedAndNotEncrypted (1),  
                        EncryptedButNotSigned (2),  
                        signedAndEncrypted (3)  
                      } 
```

Following from the MSP Specification [2], the Type `UTCTime` is encoded using the form year, month, day, hours, minutes followed by a "Z". The time is always expressed in Greenwich Mean Time.
The target ProtectionRequest is borrowed from the SecurityParameters defined in X.511 [6] for the DAP. It may appear only in a directory operation argument (a request for an operation to be carried out). It indicates the user’s preference regarding the degree of protection to be provided to the corresponding directory operation result. Instead of two levels (none and signed) as defined in [6], four levels are provided here: none (no protection requested), signedAndNotEncrypted (the MSP directory agent (MSPDA) is requested to sign but not encrypt the result), EncryptedButNotSigned (the MSPDA is requested to encrypt but not sign the result, the default), and SignedAndEncrypted (the MSPDA is requested to sign and encrypt the result). Depending on the authorisation information of the user and the DSA (possibly based on the PRBAC facility of MSP), the MSPDA could ignore none and it encrypts but does not sign the result. For the same reason, the MSPDA could ignore signedAndNotEncrypted and it signs and encrypts the result.

The DirectoryBindArgumentExtn should be used by the MSP user agent (MSPUA) and it also should be recognised by the target MSP directory agent (MSPDA). To protect the directoryBindArgument, it is enough to provide the necessary security services (in terms of signature generation and encryption) to the DirectoryBindArgumentExtnValue. The next section will show how these security services are provided.

Similarly, the DirectoryBindResultExtn and the DirectoryBindErrorExtn may be defined as stated in the appendix (Section 8).
4 MSP-Supported Directory Bind Operations

This section considers the syntax of only the directory bind operations, as our main example of the application of MSP for supporting the typical directory operations. The detail of the MSP support for the other operations including read, search, and modify operations are provided in the appendix (Section 8).

For the bind operation argument or result, this section will define an instance of the Msp for its protection. In view of security services provided by the MSP, the original optional security-specific components in the standardised arguments and results may not be required.

The directory binding is initiated by the DUA. Using the binding, the DUA automatically may bind to the home DSA when the user first logs on to the application, or when he/she makes the first directory request. The abstract syntax of the operation, called directoryBind, is given in [6]. It provides the DirectoryBindArgument, DirectoryBindResult and DirectoryBindError.

4.1 MSP-Directory Bind Argument

The MSP-supported DirectoryBindArgument is denoted by MSP-DirectoryBindArgument and it is defined as follows.

The MSP-DirectoryBindArgument is considered as an instance of the Msp and it should be originated by a MSPUA only. Hence, it inherits the abstract syntax structure of the Msp. For its definition, it is assumed that the user must select the security service (user identity authentication) from the MSPUA to support the DirectoryBindArgument.

The MSP-DirectoryBindArgument is a sequence of originatorSecurityData, signatureBlock, recipientSecurityData, optional contentDescription, directoryBindArgumentExt, optional mspSequenceSignatureAlgorithm, and optional mspEncapsulatedContentCertificate. The optional mControlInformation and mEncapsulatedContent of the original Msp do not appear in the MSP-DirectoryBindArgument since there is no requirement for a mailing list or any other information to be carried from the DUA to the DSA. The optional contentDescription, mspSequenceSignatureAlgorithm, and mspEncapsulatedCertificate may be used in situations as defined in the MSP specification [2].

```
MSP-DirectoryBindArgument ::= SEQUENCE {
    originatorSecurityData OriginatorSecurityData,
    signatureBlock SignatureBlock,
    recipientSecurityData SET OF PerRecipientToken,
    contentDescription TeletexString OPTIONAL,
}
```
4.1.1 Originator Security Data

The OriginatorSecurityData argument is exactly that as defined in the MSP specification [2] except that the mKeyToken argument is not required.

4.1.2 Signature Block

The SignatureBlock contains information used to provide non-repudiation with proof of origin. The MSPUA calculates a digital signature, which consists of signing a hash using the originator’s private cryptographic signature material. The hash is calculated by first generating a complete hash over the (original unprotected) extnValue (namely the DirectoryBindArgumentExtnValue) of the directoryBindArgumentExtn. This hash is the extnhash, which is placed in the RecipientKeyToken, if extension confidentiality is invoked. A second hash value is calculated over the concatenation of the extnhash and the signatureInformation within the ControlInformation field. This second hash is used as the input to the signature algorithm, which the MSPUA signs and includes as the SignatureValue. Similar to the case for messages described in [2], whenever the MSPUA applies confidentiality to the extnValue (namely the DirectoryBindArgumentExtnValue), it must apply confidentiality to the signature value of the extnValue.

SignatureBlock ::= SEQUENCE { signatureAlgorithm AlgorithmIdentifier, signatureValue SignatureValue, controlInformation ControlInformation, signatureCertificate CertificationPath OPTIONAL }

The SignatureValue is the same as that defined in [2]. It contains either the plain or the encrypted signature value of the DirectoryBindArgumentExtnValue.

As defined in [2], the ControlInformation is either the SignatureInformation that contains additional information from the originator including receipt requests, or the ReceiptInformation that contains information from the recipient used to identify the message for which the recipient has returned the receipt. The MSP-DirectoryBindArgument (which is what is being addressed in this subsection) is not a receipt. Hence, the ControlInformation can be only the SignatureInformation. Furthermore, there is no need for the MSP-DirectoryBindArgument to include a receipt request. Hence, the SignatureInformation consists of just the sequence of the optional encapsulatedContentType and the noReceipt ReceiptsIndicator. The encapsulatedContentType is id-empty-content (ID ::= (id-formats 2)) because the MSp-
DirectoryBindArgument does not have an encapsulatedContent. The noReceipt is the integer 0 corresponding to the case that no receipt is required.

4.1.3 **Per Recipient Token**

There is only one recipient (namely the target DSA) for the MSP-DirectoryBindArgument. Hence, there is only one PerRecipientToken in the SET OF PerRecipientToken. The PerRecipientToken is a sequence of the Tag and the ProtectedRecipientToken.

\[
\text{PerRecipientToken} ::= \text{SEQUENCE} \begin{cases} 
\text{tag} & \text{Tag,} \\
\text{protectedRecipientKeyToken} & \text{ProtectedRecipientKeyToken} \end{cases}
\]

\[
\text{ProtectedRecipientKeyToken} ::= \text{OCTET STRING}
\]

-- Protected form of RecipientKeyToken

The Tag is generated by the MSPUA as described in the MSP Specification [2].

The ProtectedRecipientKeyToken is the protected form of RecipientKeyToken. The RecipientKeyToken contains the extnKey that is used to protect the extnValue (namely the DirectoryBindArgumentExtnValue). The extnHash is calculated over the unprotected DirectoryBindArgumentExtnValue.

\[
\text{RecipientKeyToken} ::= \text{SEQUENCE} \begin{cases} 
\text{extnKey} & \text{OCTET STRING,} \\
\text{extnHash} & \text{OCTET STRING,} \\
\text{signatureBlockIndicator} & \text{BOOLEAN,} \\
\text{additionalSecurityInfoIndicator} & \text{BOOLEAN,} \\
\text{encapsulatedContentType} & \text{ContentType,} \\
\text{securityLabel} & \text{SecurityLabel} \end{cases}
\]

The signatureBlockIndicator is set to TRUE if the signatureBlock is present in the MSP header. Hence, for the MSP-DirectoryBindArgument, this indicator is TRUE. The additionalSecurityInfoIndicator is set to TRUE if the additionalSecurityInfo is present in OriginatorSecurityData. The encapsulatedContentType is set to the id-empty-content (ID := (id-formats 2)). The SecurityLabel is defined as a set of security-policy-identifier, security-classification, privacy-mark, and security-categories as specified in [2].

4.1.4 **Directory Bind Argument Extension**

The DirectoryBindArgumentExtn is the field that actually characterises the MSP-DirectoryBindArgument. As defined in Section 3.4, the DirectoryBindArgumentExtn contains the DirectoryBindArgumentExtnValue, which is a sequence of the DirectoryBindArgument, the UTCTime, and the ProtectionRequest.
Without the optional security-specific credentials argument, the DirectoryBindArgument would consist of just the Versions argument. In the standard [6], the credentials argument in the DirectoryBindArgument is designed to allow the target DSA to establish the identity of the user who is making the request through the DUA. Since the MSP-DirectoryBindArgument has achieved the same through the originatorSecurityData, signatureBlock, and recipientSecurityData fields as described in the previous subsections, the credentials may not be required in the DirectoryBindArgument. Hence, the DirectoryBindArgument may consist of the Versions argument only.

DirectoryBindArgument ::= Versions DEFAULT {v1}
Versions ::= BIT STRING {v1 (0)}

4.2 MSP-Directory Bind Result

The MSP-supported DirectoryBindResult is denoted by MSP-DirectoryBindResult and it is defined as follows.

The MSP-DirectoryBindResult also is considered as an instance of the Msp. Hence, it inherits the abstract syntax structure of the Msp. It should be originated only from the MSP directory agent (MSPDA) associated with a DSA. The MSPDA does not send a MSP-DirectoryBindResult unless

1. it has accepted a MSP-DirectoryBindArgument earlier;
2. the DirectoryBindArgument has been passed to the DSA; and
3. the DSA decides that the bind request succeeds.

In defining the MSP-DirectoryBindResult, it is assumed that the MSPDA must select the security service (directory identity authentication) to support the DirectoryBindResult. From its definition in [6], the DirectoryBindResult has the same syntax as the DirectoryBindArgument. Hence, it has the same security requirement as the DirectoryBindArgument.

MSP-DirectoryBindResult = MSP ::= SEQUENCE {
  originatorSecurityData OriginatorSecurityData,
  signatureBlock SignatureBlock,
  recipientSecurityData SET OF PerRecipientToken,
  contentDescription TeletextString OPTIONAL,
  directoryBindResultExt DirectoryBindResultExt,
  mspSequenceSignatureAlgorithm AlgorithmIdentifier OPTIONAL,
  mspSequenceSignatureCertificate CertificatePath OPTIONAL
}

The requirements and the processing of the fields (OriginatorSecurityData, SignatureBlock, and PerRecipientToken) of the MSP-DirectoryBindResult are exactly those of the MSP-DirectoryBindArgument (described in Sections 4.1.1, 4.1.2, and 4.1.3).
For the DirectoryBindResultExtn, it is defined as an instance of the Extension (Section 3.4).

DirectoryBindResultExtn Extension ::= { 
  DirectoryBindResultExtnIdentifier, 
  TRUE, 
  DirectoryBindResultExtnValue 
}

The DirectoryBindResultExtnValue is a sequence of the DirectoryBindResult and the UTCTime. Just as the case for the DirectoryBindArgument, the DirectoryBindResult consists of only the Versions.

DirectoryBindResultExtnValue ::= SEQUENCE { 
  directoryBindResult DirectoryBindResult, 
  dirDate UTCTime } 

4.3 MSP-Directory Bind Errors
The MSP-supported DirectoryBindError is denoted by MSP-DirectoryBindError and it is defined as follows.

The MSP-DirectoryBindError also is considered as an instance of the Msp. It should be originated only from the MSP directory agent (MSPDA) associated with a DSA. The MSPDA does not send a MSP-DirectoryBindError unless

1. it has accepted a MSP-DirectoryBindArgument earlier;
2. the DirectoryBindArgument has been passed to the DSA; and
3. the DSA decides that the bind request fails because of some service errors.

Since the MSP protection is available, the DSA need not address any security errors associated with the bind operation. These security errors are addressed by the MSPDA. As a result, if a security error occurs, it is indicated at the MSP level (between the MSPDA and the MSPUA).

In defining the MSP-DirectoryBindError, it is assumed that the MSPDA must select the security service (directory identity authentication) to support the DirectoryBindError.

MSP-DirectoryBindError MSP ::= SEQUENCE { 
  orignatorSecurityData OriginatorSecurityData, 
  signatureBlock SignatureBlock, 
  recipientSecurityData SET OF PerRecipientToken, 
  contentDescription TeletexString OPTIONAL, 
  directoryBindErrorExtn DirectoryBindErrorExtn, 
  mspSequenceSignatureAlgorithm AlgorithmIdentifier OPTIONAL, 
  mspSequenceSignatureCertificate CertificationPath OPTIONAL 
}
The requirements and the processing of the fields (OriginatorSecurityData, SignatureBlock, and PerRecipientToken) of the MSP-DirectoryBindError are exactly those of the MSP-DirectoryBindArgument (described in Sections 4.1.1, 4.1.2, and 4.1.3). For the DirectoryBindErrorExtn, it is defined as an instance of the Extension (Section 3.4).

DirectoryBindErrorExtn Extension ::= {
    DirectoryBindErrorExtnIdentifier,
    TRUE,
    DirectoryBindErrorExtnValue    }

The DirectoryBindErrorExtnValue is a sequence of the ERROR and the UTCTime.

DirectoryBindErrorExtnValue ::= SEQUENCE {
    directoryBindError ERROR,
    dirDate UTCTime    }

The ERROR is defined to be a parameter set of Versions and ServiceProblem [6]. As mentioned in the above, the directoryBindError does not address any security errors.
5 Some MSP-protected DUA-DSA Associations

This section presents some protected Directory Access Protocol (DAP) associations between a directory user agent (DUA) and a directory system agent (DSA) of various classification levels that become possible because of the security protection provided by the MSP.

5.1 DUA-DSA System High

Suppose that both the DUA and DSA are of the same classification level and they belong to the same system high network. The DUA-DSA association is indicated by DAP (A) for their DAP interaction in Figure 4. As the confidentiality security service for the DAP is covered by that of the system high network, the MSPUA and MSPDA may not be required to provide this service. However, the MSPUA and MSPDA still are required to provide the other security services (for example, for auditing or accounting purpose).

In addition to the DUAs belonging to the same Secret enclave, the DSA also may accept the directory binding request from a DUA of higher classification level. For example, in Figure 4, the Top Secret DUA may initiate a binding request to the Secret DSA via the DAP (B) in order to query or update the DIB stored in the DSA. Similarly, the Secret DUA may initiate a binding request to the DSA located at the Combined Force HQ via the DAP (H).

5.2 Remote DUA-DSA System High

The Secret DUA may initiate a binding request to a remote Secret DSA via the DAP (C) in Figure 4. If the operation arguments and results of the DAP need to be transported via some networks of lower classification level, then the MSPUA and MSPDA must provide the confidentiality as well as the other security services.

5.3 High DUA and Low DSA

The Secret DUA may initiate a binding request to

- a Confidential DSA via the DAP (D) in Figure 4;
- a Restricted DSA via the DAP (E) in Figure 4;
- an Unclassified DSA via the DAP (F) in Figure 4;
- a DSA belonging to another Government department via the DAP (G) in Figure 4;
- a DSA belonging to the Combined Force HQ via the DAP (H) in Figure 4.

All directory arguments and results of these DAP associations are required to depart from the Secret enclave through only the MSP gateway (MSPGW). The same is true for the DAP (B) and the DAP (C) also. In this case, the mspSequenceSignatureAlgorithm field within the Msp is used by the MSPUA at the Secret DUA, the MSPDA at the Secret DSA, and the MSPGW to identify the algorithm that the MSPUA and the MSPDA use to seal the Msp that they generate. This seal then can be verified by the MSPGW before it allows the validated
Msp to depart from the Secret enclave. According to the MSP specification [2], the Msp can be sealed by calculating a one-way hash on it and then sealing the resulting value. The sealed Msp, denoted by SIGNED Msp in [2], is a two-component sequence consisting of the Msp (which is the first component) and its seal value (which is the second).

![Diagram of MSP-enabled DUA-DSA Interactions](image)

Figure 4 Various MSP-enabled DUA-DSA Interactions

### 5.4 Classified DUA and Unclassified DSA

A DAP association between a classified DUA and an unclassified DSA is possible via the DAP (F) in Figure 4. The MSPUA at the DUA must generate the seal for the Msp which consists of only the Extension field. This Extension field contains exactly one of the directory operation arguments created by the DUA as described in Section 3.4. If the sealed Msp is validated successfully by the MSPGW, the MSPGW extracts the directory argument contained in the extnValue of the Extension, and allows it to depart from the enclave of the DUA. Furthermore, depending on the requirement of the unclassified DSA, the directory operation arguments and results could contain the Credentials or the SecurityParameters.
(defined in the X.511 standard [6]) in order to achieve the minimal security services for the unclassified DSA.

5.5 **Illegal Associations**

With only the security services provided by the MSP encapsulation of directory operation arguments and results (described in this paper), the following associations still remain illegal.

An association between a low DUA and a high DSA is not legal. The DSA must require the finer grain security services which are applicable at the individual directory entry level in order to allow access only to those entries which can be released to the low DUA. The use of the PRBAC and LRBAC facilities of MSP are only good enough for controlling a user's access to a directory system. As explained in Section 3, this is achieved through the association with the user's directory bind, read, search, and modify operation requests.

Typically in a combined operation scenario, the DUA at the Combined Force HQ would processes national information which has been released to friendly nations. Hence, an association between a DUA located at Combined Force HQ and a Secret DSA (Figure 4) also is not legal. The DSA also must require the finer grain security services which are applicable at the individual directory entry level in order to allow access only to those entries which can be released to friendly nations.
This is a blank page.
6 Conclusion

This paper has outlined an option to use the Message Security Protocol (MSP) for providing the security service support (including authentication, non-repudiation with proof, confidentiality, and partition and local rule based access control) to the Directory Access Protocol (DAP). The DAP (which specifies the directory operation argument and result syntax) is used by the Directory User Agent (DUA) to request an operation, and by the Directory System Agent (DSA) to respond to the request. This paper suggests (in Figure 3) that

- a MSP user agent (MSPUA) (for the directory user) resides between the DUA and the OSI application services including the ACSE and ROSE; and
- a MSP directory agent (MSPDA) (for the directory) resides between the DSA and the OSI application services including the ACSE and ROSE.

To provide the above security services, the MSPUA and MSPDA use the protocol Msp to communicate between themselves. The Msp is a sequence of several optional fields including: originatorSecurityData, signatureBlock, recipientSecurityData, contentDescription, Extensions, mspSequenceSignatureAlgorithm, and mspSequenceSignatureCertificate.

The MSPUA uses an instance of the Msp to encapsulate a directory argument of the DUA as an extension value within the Extensions field. Upon its arrival, the MSPDA processes the Msp instance and checks that the originator has applied the claimed security services. If the check succeeds, it then passes the directory argument (found in the encapsulation) to the DSA. When the DSA responds to the directory argument, the MSPDA similarly encapsulates the responding directory result of the DSA in an instance of the Msp which then is sent to the MSPUA.

For each of the directory operation arguments and results of the DAP, it is required that an instance of the Msp is defined for its encapsulation. The detail of these Msp instances is given in the appendix (Section 8). We have seen in Section 4 that the encapsulation occurs at the Extensions field of the Msp. We also have argued in Section 3 that the Extensions field is more flexible to use and adopt because it is required only that a new Extension is registered (so that an extnID is assigned), along with its syntax and the elements of procedure for the originator and recipient [2]. Finally, we also have discussed some protected DAP associations (that become possible because of the security protection provided by the MSP), as well as the current limitations in Section 5. The option of using the MSP therefore may be viewed as an interim step in the migration towards the fine grain directory entry access control security services.
This is a blank page.
7 References


This is a blank page.
8 Appendix: The MSP-Supported Directory Operations

This appendix considers the syntax of the directory bind, read, search, and modify operations. For each of the operation arguments and results, an instance of the Msp is defined for its protection. In view of security services provided by the MSP, the original optional security-specific components in the standardised arguments and results may not be required.

8.1 Binding Operations

The directory binding is initiated by the DUA. Using the binding, the DUA automatically may bind to the home DSA when the user first logs on to the application, or when he/she makes the first directory request. The abstract syntax of the operation, called directoryBind, is given in [6]. It provides the DirectoryBindArgument, DirectoryBindResult and DirectoryBindError.

8.1.1 MSP-Directory Bind Argument

The MSP-supported DirectoryBindArgument is denoted by MSP-DirectoryBindArgument and it is defined as follows.

The MSP-DirectoryBindArgument is considered as an instance of the Msp and it should be originated by a MSPUA only. Hence, it inherits the abstract syntax structure of the Msp. For its definition, it is assumed that the user must select the security service (user identity authentication) from the MSPUA to support the DirectoryBindArgument.

MSP-DirectoryBindArgument Msp ::= SEQUENCE {  
  originatorSecurityData OriginatorSecurityData,  
  signatureBlock SignatureBlock,  
  recipientSecurityData SET OF PerRecipientToken,  
  contentDescription TeletexString OPTIONAL,  
  directoryBindArgumentExtn DirectoryBindArgumentExtn,  
  mspSequenceSignatureAlgorithm AlgorithmIdentifier OPTIONAL,  
  mspSequenceSignatureCertificate CertificationPath OPTIONAL  }  

The requirements and the processing of the fields (OriginatorSecurityData, SignatureBlock, and PerRecipientToken) of the MSP-DirectoryBindArgument have been described in Sections 4.1.1, 4.1.2, and 4.1.3. For the DirectoryBindArgumentExtn, it is defined as an instance of the Extension (Section 3.4).

DirectoryBindArgumentExtn Extension ::= {  
  BindArgumentExtnID,  
  TRUE,  
  DirectoryBindArgumentExtnValue  }  

DirectoryBindArgumentExtnValue ::= SEQUENCE {  

UNCLASSIFIED 27
directoryBindArgument DirectoryBindArgument,
dirDate UTCTime,
target ProtectionRequest OPTIONAL }

ProtectionRequest ::= INTEGER {
    none (0),
  signedAndNotEncrypted (1),
  EncryptedButNotSigned (2),
  signedAndEncrypted (3) }

The DirectoryBindArgument is exactly that as defined in X.511 [6].

8.1.2 MSP-Directory Bind Result

In defining the MSP-DirectoryBindResult, it is assumed that the MSPDA must select the
security service (directory identity authentication) to support the DirectoryBindResult.
From its definition in [6], the DirectoryBindResult has the same syntax as the
DirectoryBindArgument. Hence, it has the same security requirement as the
DirectoryBindArgument.

MSP-DirectoryBindResult MSP ::= SEQUENCE {
  originatorSecurityData OriginatorSecurityData,
  signatureBlock SignatureBlock,
  recipientSecurityData SET OF PerRecipientToken,
  contentDescription TeletextString OPTIONAL,
  directoryBindResultExt
  directoryBindResultExt OPTIONAL,
  mspSequenceSignatureAlgorithm AlgorithmIdentifier OPTIONAL,
  mspSequenceSignatureCertificate CertificationPath OPTIONAL }

The requirements and the processing of the fields (OriginatorSecurityData,
SignatureBlock, and PerRecipientToken) of the MSP-DirectoryBindResult are exactly
those of the MSP-DirectoryBindArgument (described in Sections 4.1.1, 4.1.2, and 4.1.3).
For the DirectoryBindResultExt, it is defined as an instance of the Extension (Section
3.4).

DirectoryBindResultExt Extension ::= {
  DirectoryBindResultExtIdentifier, TRUE,
  DirectoryBindResultExtValue }

The DirectoryBindResultExtValue is a sequence of the DirectoryBindResult and the
UTCTime. Just as the case for the DirectoryBindArgument, the DirectoryBindResult
consists of only the Versions.

DirectoryBindResultExtValue ::= SEQUENCE {
  directoryBindResult DirectoryBindResult,
  dirDate UTCTime )
The DirectoryBindResult is exactly that as defined in X.511 [6].

8.1.3 MSP-Directory Bind Errors

In defining the MSP-DirectoryBindError, it is assumed that the MSPDA must select the security service (directory identity authentication) to support the DirectoryBindError.

MSP-DirectoryBindError  MSP ::= SEQUENCE {  
  originatorSecurityData OriginatorSecurityData,  
  signatureBlock SignatureBlock,  
  recipientSecurityData SET OF PerRecipientToken,  
  contentDescription TeletexString OPTIONAL,  
  directoryBindErrorExtn DirectoryBindErrorExtn,  
  mspSequenceSignatureAlgorithm AlgorithmIdentifier OPTIONAL,  
  mspSequenceSignatureCertificate CertificationPath OPTIONAL  
}

The requirements and the processing of the fields (OriginatorSecurityData, SignatureBlock, and PerRecipientToken) of the MSP-DirectoryBindError are exactly those of the MSP-DirectoryBindArgument (described in Sections 4.1.1, 4.1.2, and 4.1.3). For the DirectoryBindErrorExtn, it is defined as an instance of the Extension (Section 3.4).

DirectoryBindErrorExtn Extension ::= {  
  DirectoryBindErrorExtnIdentifier,  
  TRUE,  
  DirectoryBindErrorExtnValue  
}

The DirectoryBindErrorExtnValue is a sequence of the ERROR and the UTCTime.

DirectoryBindErrorExtnValue ::= SEQUENCE {  
  directoryBindError ERROR,  
  dirDate UTCTime  
}

The ERROR is defined to be a parameter set of Versions and ServiceProblem [6]. As mentioned in the above, the directoryBindError does not address any security errors.

8.2 Directory Read Operations

The directory read operation consists of the ReadArgument, ReadResult, and ERRORS. The ReadArgument is used by the DUA to extract information from an explicitly identified entry. If the request in the ReadArgument succeeds, the ReadResult is used by the target DSA to return the information requested in the ReadArgument. If the request in the ReadArgument fails, the ERRORS is used by the DSA to indicates either AttributeError, NameError, ServiceError, Referral, Abandoned, or SecurityError. Denote the MSP-supported ReadArgument, ReadResult, and ERRORS by MSP-ReadArgument, MSP-ReadResult, and MSP-ERRORS respectively.
8.2.1 MSP-Read Argument

In defining the MSP-ReadArgument, it is assumed that the MSPUA must select the (non-repudiation with proof of operation request) security service to support the ReadArgument. It may select the confidentiality security service also.

MSP-ReadArgument  MSP ::= SEQUENCE  
originatorSecurityData  OriginatorSecurityData, 
signatureBlock  SignatureBlock, 
recipientSecurityData  SET OF PerRecipientToken, 
contentDescription  TeletextString OPTIONAL, 
readArgumentExtn  ReadArgumentExtn, 
mspSequenceSignatureAlgorithm  AlgorithmIdentifier OPTIONAL, 
mspSequenceSignatureCertificate  CertificationPath OPTIONAL  
}

The requirements and the processing of the fields (OriginatorSecurityData, SignatureBlock, and PerRecipientToken) of the MSP-ReadArgument are exactly those of the MSP-DirectoryBindArgument (described in Sections 4.1.1, 4.1.2, and 4.1.3). For the ReadArgumentExtn, it is defined as an instance of the Extension (Section 3.4).

ReadArgumentExtn  Extension ::=  
  ReadArgumentExtnIdentifier, 
  TRUE, 
  ReadArgumentExtnValue  
}

The ReadArgumentExtnValue is a sequence of the ReadArgument, the UTCTime, and the optional ProtectionRequest.

ReadArgumentExtnValue ::= SEQUENCE  
  readArgument  ReadArgument, 
  dirDate  UTCTime, 
  target  ProtectionRequest OPTIONAL  
}

The ReadArgument is exactly that as defined in X.511 [6].

8.2.2 MSP-Read Result

In defining the MSP-ReadResult, it is assumed that the MSPDA must select the (non-repudiation with proof of operation result) security service to support the ReadResult. It may select the confidentiality security service also.

In addition, the MSPDA may select the (non-repudiation with proof of delivery of operation result) security service. In this case, the MSPDA requests the origin MSPUA to return a receipt when the MSPUA receives the MSP-ReadResult. The ReceiptsIndicator is set to allReceipts (1) within the SignatureInformation of the ControlInformation. When the MSPUA generates the receipt for the MSPDA, it follows the procedures as specified in the MSP specification [2]. The receipt then may be transported by using the ROSE services.
MSP-ReadResult

MSP ::= SEQUENCE {
  originatorSecurityData OriginatorSecurityData,
  signatureBlock SignatureBlock,
  recipientSecurityData SET OF PerRecipientToken,
  contentDescription TeletexString OPTIONAL,
  readResultExtn ReadResultExtn,
  mspSequenceSignatureAlgorithm AlgorithmIdentifier OPTIONAL,
  mspSequenceSignatureCertificate CertificationPath OPTIONAL
}

The requirements and the processing of the fields (OriginatorSecurityData, SignatureBlock, and PerRecipientToken) of the MSP-ReadResult are exactly those of the MSP-DirectoryBindArgument (described in Sections 4.1.1, 4.1.2, and 4.1.3). For the ReadResultExtn, it is defined as an instance of the Extension (Section 3.4).

ReadResultExtn Extension ::= {
  ReadResultExtnIdentifier,
  TRUE,
  ReadResultExtnValue
}

The ReadResultExtnValue is a sequence of the of ReadResult and the UTCTime.

ReadResultExtnValue ::= SEQUENCE {
  readResult ReadResult,
  dirDate UTCTime
}

The ReadResult is exactly that as defined in X.511 [6].

8.2.3 MSP-Errors

In defining the MSP-ERRORS, it is assumed that the MSPDA may select the data-origin-authentication security service to support the ERRORS.

MSP-ERRORS MSP ::= SEQUENCE {
  originatorSecurityData OriginatorSecurityData,
  signatureBlock SignatureBlock,
  recipientSecurityData SET OF PerRecipientToken,
  contentDescription TeletexString OPTIONAL,
  errorsExtn ERRORSExn,
  mspSequenceSignatureAlgorithm AlgorithmIdentifier OPTIONAL,
  mspSequenceSignatureCertificate CertificationPath OPTIONAL
}

The requirements and the processing of the fields (OriginatorSecurityData, SignatureBlock, and PerRecipientToken) of the MSP-ERRORS are exactly those of the MSP-DirectoryBindArgument (described in Sections 4.1.1, 4.1.2, and 4.1.3). For the ERRORSExn, it is defined as an instance of the Extension (Section 3.4).

ERRORSExn Extension ::= {
  ERRORSExnIdentifier,
  TRUE,
The `ERRORSExtnValue` is a sequence of the `ERRORS` and the `UTCTime`.

\[
\text{ERRORSExtnValue} ::= \text{SEQUENCE} \{
\text{ERRORS} \{ \text{AttributeError | NameError | ServiceError | Referral | Abandoned | SecurityError} \}
\text{dirDate} \text{UTCTime} \}
\]

As mentioned in Section 8.1.3, security errors related to the MSP security services are handled at the MSP level (between MSPDA and MSPUA). However, there are other security errors which are not handled by the MSPDA and MSPUA. One such security error is the `insufficientAccessRights` defined in X.511 [6]. It is used by the DSA to indicate that the requester does not have the right to a directory entry (which is required to carry out the requested operation). The MSP does not address the access control of the individual entries. The role of MSP is only to address the security requirements of the directory operation arguments and results between the DUA and DSA.

### 8.3 Other Directory Operations

For the other directory operations: `compare`, `search`, `list`, `addEntry`, `removeEntry`, `modifyEntry`, and `modifyDN`, their arguments and results also have the corresponding MSP-supported arguments and results. They are defined in the same way as the `MSP-ReadArgument` and the `MSP-ReadResult` are defined in Section 8.2.

#### 8.3.1 Compare Operations

##### 8.3.1.1 Compare Argument

\[
\text{MSP-CompareArgument} \quad \text{MSP} ::= \text{SEQUENCE} \{
\text{originatorSecurityData} \text{OriginatorSecurityData},
\text{signatureBlock} \text{SignatureBlock},
\text{recipientSecurityData} \text{SET OF PerRecipientToken},
\text{contentDescription} \text{TeletexString OPTIONAL},
\text{compareArgumentExtn} \text{CompareArgumentExtn},
\text{mspSequenceSignatureAlgorithm} \text{AlgorithmIdentifier OPTIONAL},
\text{mspSequenceSignatureCertificate} \text{CertificationPath OPTIONAL} \}
\]

\[
\text{CompareArgumentExtn} \text{Extension} ::= \{ \text{CompareArgumentExtnIdentifier}, \text{TRUE}, \text{CompareArgumentExtnValue} \}
\]

\[
\text{CompareArgumentExtnValue} ::= \text{SEQUENCE} \{
\text{compareArgument} \text{CompareArgument},
\text{dirDate} \text{UTCTime},
\text{target} \text{ProtectionRequest OPTIONAL} \}
\]
8.3.1.2 Compare Result

MSP-CompareResult MSP ::= SEQUENCE {
  originatorSecurityData OriginatorSecurityData,
  signatureBlock SignatureBlock,
  recipientSecurityData SET OF PerRecipientToken,
  contentDescription TeletextString OPTIONAL,
  compareResultExtn CompareResultExtn,
  mspSequenceSignatureAlgorithm AlgorithmIdentifier OPTIONAL,
  mspSequenceSignatureCertificate CertificationPath OPTIONAL }

CompareResultExtn Extension ::= {
  CompareResultExtnIdentifier,
  TRUE,
  CompareResultExtnValue
}

CompareResultExtnValue ::= SEQUENCE {
  compareResult CompareResult,
  dirDate UTCTime
}

8.3.2 List Operations

8.3.2.1 List Argument

MSP-ListArgument MSP ::= SEQUENCE {
  originatorSecurityData OriginatorSecurityData,
  signatureBlock SignatureBlock,
  recipientSecurityData SET OF PerRecipientToken,
  contentDescription TeletextString OPTIONAL,
  listArgumentExtn ListArgumentExtn,
  mspSequenceSignatureAlgorithm AlgorithmIdentifier OPTIONAL,
  mspSequenceSignatureCertificate CertificationPath OPTIONAL }

ListArgumentExtn Extension ::= {
  ListArgumentExtnIdentifier,
  TRUE,
  ListArgumentExtnValue
}

ListArgumentExtnValue ::= SEQUENCE {
  listArgument ListArgument,
  dirDate UTCTime,
  target ProtectionRequest OPTIONAL
}

8.3.2.2 List Result

MSP-ListResult MSP ::= SEQUENCE {
  originatorSecurityData OriginatorSecurityData,
  signatureBlock SignatureBlock,
  recipientSecurityData SET OF PerRecipientToken,
  contentDescription TeletextString OPTIONAL,
  listResultExtn ListResultExtn,
  mspSequenceSignatureAlgorithm AlgorithmIdentifier OPTIONAL.
8.3.3 Search Operations

8.3.3.1 Search Argument

MSP-SearchArgument

MSP := SEQUENCE {  
  originatorSecurityData OriginatorSecurityData,  
  signatureBlock SignatureBlock,  
  recipientSecurityData SET OF PerRecipientToken,  
  contentDescription TeletexString OPTIONAL,  
  searchArgumentExtn SearchArgumentExtn,  
  mspSequenceSignatureAlgorithm AlgorithmIdentifier OPTIONAL,  
  mspSequenceSignatureCertificate CertificationPath OPTIONAL  
}

SearchArgumentExtn Extension := {  
  SearchArgumentExtnIdentifier,  
  TRUE,  
  SearchArgumentExtnValue }  

SearchArgumentExtnValue := SEQUENCE {  
  searchArgument SearchArgument,  
  dirDate UTCTime,  
  target ProtectionRequest OPTIONAL  
}

8.3.3.2 Search Result

MSP-SearchResult

MSP := SEQUENCE {  
  originatorSecurityData OriginatorSecurityData,  
  signatureBlock SignatureBlock,  
  recipientSecurityData SET OF PerRecipientToken,  
  contentDescription TeletexString OPTIONAL,  
  searchResultExtn SearchResultExtn,  
  mspSequenceSignatureAlgorithm AlgorithmIdentifier OPTIONAL,  
  mspSequenceSignatureCertificate CertificationPath OPTIONAL  
}

SearchResultExtn Extension := {  
  SearchResultExtnIdentifier,  
  TRUE,  
  SearchResultExtnValue }  

SearchResultExtnValue := SEQUENCE {  
  searchResult SearchResult,  
  dirDate UTCTime  
}
8.3.4 Add Entry Operations

8.3.4.1 Add Entry Argument

MSP-AddEntryArgument \( \text{MSP} ::= \text{SEQUENCE} \) 
\[ \begin{align*}
\text{originatorSecurityData} & : \text{OriginatorSecurityData}, \\
\text{signatureBlock} & : \text{SignatureBlock}, \\
\text{recipientSecurityData} & : \text{SET OF PerRecipientToken}, \\
\text{contentDescription} & : \text{TeletextString OPTIONAL}, \\
\text{addEntryArgumentExtn} & : \text{AddEntryArgumentExtn}, \\
\text{mspSequenceSignatureAlgorithm} & : \text{AlgorithmIdentifier OPTIONAL}, \\
\text{mspSequenceSignatureCertificate} & : \text{CertificationPath OPTIONAL} \\
\end{align*} \]

AddEntryArgumentExtn Extension ::=
\[ \begin{align*}
\text{AddEntryArgumentExtnIdentifier}, & \\
\text{TRUE}, & \\
\text{AddEntryArgumentExtnValue} & \\
\end{align*} \]

AddEntryArgumentExtnValue ::= \( \text{SEQUENCE} \)
\[ \begin{align*}
\text{addEntryArgument} & : \text{AddEntryArgument}, \\
\text{dirDate} & : \text{UTCTime}, \\
\text{target} & : \text{ProtectionRequest OPTIONAL} \\
\end{align*} \]

8.3.4.2 Add Entry Result

MSP-AddEntryResult MSP ::= \( \text{SEQUENCE} \)
\[ \begin{align*}
\text{originatorSecurityData} & : \text{OriginatorSecurityData}, \\
\text{signatureBlock} & : \text{SignatureBlock}, \\
\text{recipientSecurityData} & : \text{SET OF PerRecipientToken}, \\
\text{contentDescription} & : \text{TeletextString OPTIONAL}, \\
\text{addEntryResultExtn} & : \text{AddEntryResultExtn}, \\
\text{mspSequenceSignatureAlgorithm} & : \text{AlgorithmIdentifier OPTIONAL}, \\
\text{mspSequenceSignatureCertificate} & : \text{CertificationPath OPTIONAL} \\
\end{align*} \]

AddEntryResultExtn Extension ::= 
\[ \begin{align*}
\text{AddEntryResultExtnIdentifier}, & \\
\text{TRUE}, & \\
\text{AddEntryResultExtnValue} & \\
\end{align*} \]

AddEntryResultExtnValue ::= \( \text{SEQUENCE} \)
\[ \begin{align*}
\text{addEntryResult} & : \text{AddEntryResult}, \\
\text{dirDate} & : \text{UTCTime} \\
\end{align*} \]

8.3.5 Remove Entry Operations

8.3.5.1 Remove Entry Argument

MSP-RemoveEntryArgument MSP ::= \( \text{SEQUENCE} \)
\[ \begin{align*}
\text{originatorSecurityData} & : \text{OriginatorSecurityData}, \\
\text{signatureBlock} & : \text{SignatureBlock}, \\
\text{recipientSecurityData} & : \text{SET OF PerRecipientToken}, \\
\text{contentDescription} & : \text{TeletextString OPTIONAL} \\
\end{align*} \]
8.3.5.2 Remove Entry Result

8.3.6 Modify Entry Operations

8.3.6.1 Modify Entry Argument
8.3.6.2 Modify Entry Result

MSP-ModifyEntryResult MSP ::= SEQUENCE {
  originatorSecurityData OriginatorSecurityData,
  signatureBlock SignatureBlock,
  recipientSecurityData SET OF PerRecipientToken,
  contentDescription TeletextString OPTIONAL,
  modifyEntryResultExtn ModifyEntryResultExtn,
  mspSequenceSignatureAlgorithm AlgorithmIdentifier OPTIONAL,
  mspSequenceSignatureCertificate CertificationPath OPTIONAL }

ModifyEntryResultExtn Extension ::= {
  ModifyEntryResultExtnIdentifier,
  TRUE,
  ModifyEntryResultExtnValue }

ModifyEntryResultExtnValue ::= SEQUENCE {
  modifyEntryResult ModifyEntryResult,
  dirDate UTCTime }

8.3.7 Modify DN Operations

8.3.7.1 Modify DN Argument

MSP-ModifyDNArgument MSP ::= SEQUENCE {
  originatorSecurityData OriginatorSecurityData,
  signatureBlock SignatureBlock,
  recipientSecurityData SET OF PerRecipientToken,
  contentDescription TeletextString OPTIONAL,
  modifyDNArgumentExtn ModifyDNArgumentExtn,
  mspSequenceSignatureAlgorithm AlgorithmIdentifier OPTIONAL,
  mspSequenceSignatureCertificate CertificationPath OPTIONAL }

ModifyDNArgumentExtn Extension ::= {
  ModifyDNArgumentExtnIdentifier,
  TRUE,
  ModifyDNArgumentExtnValue }

ModifyDNArgumentExtnValue ::= SEQUENCE {
  modifyDNArgument ModifyDNArgument,
  dirDate UTCTime,
  target ProtectionRequest OPTIONAL }

8.3.7.2 Modify DN Result

MSP-ModifyDNResult MSP ::= SEQUENCE {
  originatorSecurityData OriginatorSecurityData,
  signatureBlock SignatureBlock,
recipientSecurityData
contentDescription
modifyDNResultExt
mspSequenceSignatureAlgorithm
mspSequenceSignatureCertificate

ModifyDNResultExtn  Extension ::= {
  ModifyDNResultExtnIdentifier,
  TRUE,
  ModifyDNResultExtnValue
}

ModifyDNResultExtnValue ::= SEQUENCE {
  modifyDNResult  ModifyDNResult,
  dirDate        UTCTime   }

Distribution

DEPARTMENT OF DEFENCE

Science and Technology

Defence Science and Technology Organisation Central

<table>
<thead>
<tr>
<th>Function</th>
<th>Copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chief Defence Scientist and members of the</td>
<td>1 copy</td>
</tr>
<tr>
<td>DSTO Central Office Executive</td>
<td></td>
</tr>
<tr>
<td>Counsellor, Defence Science, London</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Counsellor, Defence Science, Washington</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Senior Defence Scientific Adviser</td>
<td></td>
</tr>
<tr>
<td>Scientific Adviser POLCOM</td>
<td>1 copy</td>
</tr>
</tbody>
</table>

Aeronautical & Maritime Research Laboratory

<table>
<thead>
<tr>
<th>Function</th>
<th>Copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Director Aeronautical &amp; Maritime Research Laboratory</td>
<td>1 copy</td>
</tr>
</tbody>
</table>

Electronics & Surveillance Research Laboratory

<table>
<thead>
<tr>
<th>Function</th>
<th>Copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chief Information Technology Division</td>
<td>1 copy</td>
</tr>
<tr>
<td>Chief Electronic Warfare Division</td>
<td></td>
</tr>
<tr>
<td>Chief Guided Weapons Division</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Chief Communications Division</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Chief Land, Space and Optoelectronics Division</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Chief High Frequency Radar Division</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Chief Microwave Radar Division</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Research Leader Command &amp; Control and Intelligence Systems</td>
<td>1 copy</td>
</tr>
<tr>
<td>Research Leader Military Computing Systems</td>
<td></td>
</tr>
<tr>
<td>Research Leader Command, Control and Communications</td>
<td>1 copy</td>
</tr>
<tr>
<td>Research Leader Military Information Networks</td>
<td>1 copy</td>
</tr>
<tr>
<td>Research Leader Secure Communications</td>
<td>1 copy</td>
</tr>
<tr>
<td>Head Human Systems Integration Group</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Executive Officer (ITD)</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Head Software Engineering Group</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Head Trusted Computer Systems Group</td>
<td>1 copy</td>
</tr>
<tr>
<td>Head Command Support Systems Group</td>
<td>1 copy</td>
</tr>
<tr>
<td>Head Intelligence Systems Group</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Head Systems Simulation and Assessment Group</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Head Exercise Analysis Group</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Head C3I Systems Engineering Group</td>
<td>1 copy</td>
</tr>
<tr>
<td>Head Computer Systems Architecture Group</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Head Information Management Group</td>
<td>1 copy</td>
</tr>
<tr>
<td>Head Information Acquisition &amp; Processing Group</td>
<td>Cont Sht</td>
</tr>
<tr>
<td>Head Advanced Computer Capabilities Group</td>
<td>Cont Sht</td>
</tr>
</tbody>
</table>
Navy
- Navy Scientific Adviser
  1 copy

Army
- Scientific Adviser, Army
  1 copy

Air Force
- Air Force Scientific Adviser
  1 copy

Forces Executive
- Director General Force Development (Joint)
  1 copy
- Director General Force Development (Land)
  1 copy
- Director General Force Development (Air)
  1 copy
- Director General Force Development (Sea)
  1 copy
- Director General Joint Communications and Electronics
  1 copy
- Deputy Director Network Systems
  1 copy

Acquisition and Logistics
- Director General Information Management and Communications Engineering
  1 copy
- Director General Joint Projects Management
  1 copy

Strategy and Intelligence
- Assistant Secretary Scientific Analysis
  1 copy
- Assistant Secretary Information Security
  1 copy

LIBRARIES AND INFORMATION SERVICES
- Australian Government Publishing Service
  1 copy
- Defence Central Library, Technical Reports Centre
  1 copy
- Manager, Document Exchange Centre, (for retention)
  1 copy
- Defense Technical Information Service, United States
  2 copies
- Defence Research Information Centre, United Kingdom
  2 copies
- Director Scientific Information Services, Canada
  1 copy
- Library, Ministry of Defence, New Zealand
  1 copy
- National Library of Australia
  1 copy
- Defence Science and Technology Organisation Salisbury, Research Library
  2 copies
- Library Defence Signals Directorate Canberra
  1 copy
- British Library Document Supply Centre
  1 copy
UNCLASSIFIED

Parliamentary Library of South Australia
1 copy
The State Library of South Australia
1 copy

SPARES

Defence Science and Technology Organisation Salisbury,
Research Library
6 copies

UNCLASSIFIED
This is a blank page.
**DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION**

**DOCUMENT CONTROL DATA**

<table>
<thead>
<tr>
<th>1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)</td>
<td></td>
</tr>
<tr>
<td>Document</td>
<td>(U)</td>
</tr>
<tr>
<td>Title</td>
<td>(U)</td>
</tr>
<tr>
<td>Abstract</td>
<td>(U)</td>
</tr>
<tr>
<td>4. AUTHOR(S)</td>
<td>M.K.F. Lai</td>
</tr>
<tr>
<td>5. CORPORATE AUTHOR</td>
<td>Electronics and Surveillance Research Laboratory PO Box 1500 Salisbury SA 5108</td>
</tr>
<tr>
<td>6a. DSTO NUMBER</td>
<td>DSTO-RR-0063</td>
</tr>
<tr>
<td>6b. AR NUMBER</td>
<td>AR-009-444</td>
</tr>
<tr>
<td>6c. TYPE OF REPORT</td>
<td>Research Report</td>
</tr>
<tr>
<td>7. DOCUMENT DATE</td>
<td>January 1996</td>
</tr>
<tr>
<td>8. FILE NUMBER</td>
<td>N9505/10/14</td>
</tr>
<tr>
<td>9. TASK NUMBER</td>
<td>ADF 93/256</td>
</tr>
<tr>
<td>10. TASK SPONSOR</td>
<td>HQADF</td>
</tr>
<tr>
<td>11. NO. OF PAGES</td>
<td>50</td>
</tr>
<tr>
<td>12. NO. OF REFERENCES</td>
<td>12</td>
</tr>
<tr>
<td>13. DOWNGRADING/DELIMITING INSTRUCTIONS</td>
<td>N/A</td>
</tr>
<tr>
<td>14. RELEASE AUTHORITY</td>
<td>Chief, Information Technology Division</td>
</tr>
<tr>
<td>15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT</td>
<td>APPROVED FOR PUBLIC RELEASE</td>
</tr>
<tr>
<td>OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE CENTRE, DEPT OF DEFENCE, CAMPBELL PARK OFFICES, CANBERRA ACT 2600</td>
<td></td>
</tr>
<tr>
<td>16. DELIBERATE ANNOUNCEMENT</td>
<td>No limitation</td>
</tr>
<tr>
<td>17. CASUAL ANNOUNCEMENT</td>
<td>No limitation</td>
</tr>
<tr>
<td>18. DEPTTEST DESCRIPTORS</td>
<td>Message Processing Computer Information Security</td>
</tr>
<tr>
<td></td>
<td>Secure Communication Data Security (Computer)</td>
</tr>
<tr>
<td></td>
<td>Access Control Computer Architecture</td>
</tr>
<tr>
<td>19. ABSTRACT</td>
<td>This paper is part of the documentation series produced under the HQADF sponsored task “D6: A Security Architecture for Large, Distributed Multimedia Systems”. It proposes to use the Message Security Protocol (MSP) for providing the security service support to the Directory Access Protocol (DAP). This may be viewed as an interim option for the directory service implementation before the militarised DAP is finalised and its trusted directory user agent becomes available. Specifically, this paper focuses on the necessary protocol elements of MSP for supporting the security requirements of the DAP.</td>
</tr>
</tbody>
</table>