Network Working Group
Request for Comments: 4534
Category: Standards Track

A. Colegrove
H. Harney
SPARTA, Inc.

June 2006

Group Security Policy Token v1

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

The Group Security Policy Token is a structure used to specify the security policy and configurable parameters for a cryptographic group, such as a secure multicast group. Because the security of a group is composed of the totality of multiple security services, mechanisms, and attributes throughout the communications infrastructure, an authenticatable representation of the features that must be supported throughout the system is needed to ensure consistent security. This document specifies the structure of such a token.

Table of Contents

1. Introduction 3
2. Token Creation and Receipt 4
3. The Policy Token 5
3.1. Token Identifiers 6
3.2. Registration Policy 6
3.3. Rekey Policy 7
3.4. Group Data Policy 8
4. Security Considerations 8
5. IANA Considerations 8
6. References....9
6.1. Normative References 9
6.2. Informative References 10
7. Acknowledgements 10
Appendix A. Core Policy Token ASN.1 Module 11
Appendix B. GSAKMPv1 Base Policy 13
B.1. GSAKMPv1 Registration Policy 13
B.1.1. Authorization 13
B.1.2. AccessControl 14
B.1.3. JoinMechanisms 15
B.1.3.1. alaCarte 15
B.1.3.2. suite 17
B.1.4. Transport 17
B.2. GSAKMPv1 Registration ASN.1 Module 17
B.3. GSAKMPv1 De-Registration Policy 20
B.4. GSAKMPv1 De-Registration ASN.1 Module 21
B.5. GSAKMPv1 Rekey Policy 22
B.5.1. Rekey Authorization 22
B.5.2. Rekey Mechanisms 23
B.5.3. Rekey Event Definition 23
B.5.4. Rekey Methods 24
B.5.4.1 Rekey Method NONE 24
B.5.4.2 Rekey Method GSAKMP LKH 24
B.5.5 Rekey Interval 25
B.5.6 Rekey Reliability 25
B.5.6.1 Rekey Reliability Mechanism None 25
B.5.6.2 Rekey Reliability Mechanism Resend 25
B.5.6.3 Rekey Reliability Mechanism Post 26
B.5.7 Distributed Operation Policy 26
B.5.7.1 No Distributed Operation 26
B.5.7.2 Autonomous Distributed Mode 26
B.6. GSAKMPv1 Rekey Policy ASN.1 Module 27
Appendix C. Data SA Policy 30
C.1. Generic Data Policy 30
C.2. Generic Data Policy ASN.1 Module 30

1. Introduction

The Multicast Group Security Architecture [RFC3740] defines the security infrastructure to support secure group communications. The policy token assumes this architecture in its definition. It defines the enforceable security parameters for a Group Secure Association.

The policy token is a verifiable data construct signed by the Group Owner, the entity with the authorization to create security policy. The group controllers in a group will use the policy token to ensure that the mechanisms used to secure the group are correct and to enforce the access control rules for joining members. The group members, who may contribute data to the group or access data from the group, will use the policy token to ensure that the group is owned by a trusted authority. Also, the members may want to verify that the access control rules are adequate to protect the data that the member is submitting to the group.

The policy token is specified in ASN.1 [X.208] and is to be DER [X.660] encoded. This specification ability allows the token to easily import group definitions that span different applications and environments. ASN.1 allows the token to specify branches that can be used by any multicast security protocol. Any group can use this policy token structure to specify the use of multiple protocols in securing the group.

Care was taken in this specification to provide a core level of token specificity that would allow ease of extensibility and flexibility in supporting mechanisms. This was done by using the following abstracted construct:

```
Mechanism ::= SEQUENCE {
  mechanismIdentifier  OBJECT IDENTIFIER,
  mechanismParameters OCTET STRING
}
```

This construct will allow the use of group mechanisms specified in other documents with the policy token.

The policy token is structured to reflect the MSEC Architecture layers for a Group Security Association. Each of the architectural layers is identified and given a branch in the "Core" token. This allows a high degree of flexibility for future protocol specifications at each architectural layer without the need to change the "Core" policy token, which can then act as a single point of reference for defining secure groups using any mix of protocols for any number of environments.

2. Token Creation and Receipt

At the time of group creation or whenever the policy of the group is updated, the Group Owner will create a new policy token.

To ensure authenticity of the specified policy, the Token MUST be signed by the Group Owner. The signed token MUST be in accordance with the Cryptographic Message Syntax (CMS) [RFC3852] SignedData type.

The content of the SignedData is the token itself. It is represented with the ContentType object identifier of

```
id-ct-msec-token    OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.1.1}
```

The CMS sid value of the SignerInfo, which identifies the public key needed to validate the signature, MUST be that of the Group Owner.

The signedAttrs field MUST be present. In addition to the minimally required fields of signedAttrs, the signing-time attribute MUST be present.

Upon receipt of a policy token, the recipient MUST check that

- the Group Owner, as identified by the sid in the SignerInfo, is the expected entity.
- the signing-time value is more recent than the signing-time value seen in a previously received policy token for that group, or the policy token is the first token seen by the recipient for that group.
- the processing of the signature successfully validates in accordance with RFC 3852.
- the specified security and communication mechanisms (or at least one mechanism of each choice) are supported and are in compliance with the recipient’s local policy.

3. The Policy Token

The structure of the policy token is as follows:

```
Token ::= SEQUENCE {
  tokenInfo     TokenID,
  registration  SEQUENCE OF Registration,
  rekey         SEQUENCE OF GroupMngmtProtocol,
  data          SEQUENCE OF DataProtocol
}
```

tokenInfo provides information about the instance of the Policy Token (PT).

registration provides a list of acceptable registration and de-registration policy and mechanisms that may be used to manage member-initiated joins and departures from a group. A NULL sequence indicates that the group does not support registration and de-registration of members. A member MUST be able to support at least one set of Registration mechanisms in order to join the group. When multiple mechanisms are present, a member MAY use any of the listed methods. The list is ordered in terms of Group Owner preference. A member MUST choose the highest listed mechanism that local policy supports.

rekey provides the rekey protocols that will be used in managing the group. The member MUST be able to accept one of the types of rekey messages listed. The list is ordered in terms of Group Owner preference. A member MUST choose the highest listed mechanism that local policy supports.

data provides the applications used in the communications between group members. When multiple applications are provided, the order of the list implies the order of encapsulation of the data. A member MUST be able to support all the listed applications and if any choices of mechanisms are provided per application, the member MUST support at least one of the mechanisms.

For the registration, rekey, and data fields, implementations encountering unknown protocol identifiers MUST handle this gracefully by providing indicators that an unknown protocol is among the sequence of permissible protocols. If the unknown protocol is the only allowable protocol in the sequence, then the implementation cannot support that field, and the member cannot join the group. It is a matter of local policy whether a join is permitted when an unknown protocol exists among the allowable, known protocols.

Protocols in addition to registration, rekey, and data SHOULD NOT be added to subsequent versions of this Token unless the MSEC architecture changes.

Each data field of the PT is specified further in the following sections.

3.1. Token Identifiers

tokenInfo explicitly identifies a version of the policy token for a particular group. It is defined as

```
TokenID ::= SEQUENCE {
  tokenDefVersion INTEGER (1),
  groupName       OCTET STRING,
  edition         INTEGER OPTIONAL
}
```

tokenDefVersion is the version of the Group Policy Token Specification. This specification (v1) is represented as one (1). Changes to the structure of the Group Security Policy Token will require an update to this field.

groupName is the identifier of the group and MUST be unique relative to the Group Owner.

edition is an optional INTEGER indicating the sequence number of the PT. If edition is present, group entities MUST accept a PT only when the value is greater than the last value seen in a valid PT for that group.

The type LifeDate is also defined to provide standard methods of indicating timestamps and intervals in the Tokens.

```
LifeDate ::= CHOICE {
  gt       GeneralizedTime,
  utc      UTCTime,
  interval INTEGER
}
```

3.2. Registration Policy

The registration security association (SA) is defined in the MSEC Architecture. During registration, a prospective group member and the group controller will interact to give the group member access to the keys and information it needs to join the group and participate in the group Data SA.

The de-registration piece allows a current group member to notify the Group Controller Key Server (GC/KS) that it will no longer be participating in the Data SA.

```
Registration ::= SEQUENCE {
  register    GroupMngmtProtocol,
  de-register GroupMngmtProtocol
}
```

The protocols for registration and de-registration are each specified as

```
GroupMngmtProtocol ::= CHOICE {
  none      NULL,
  supported Protocol
}

Protocol ::= SEQUENCE {
  protocol      OBJECT IDENTIFIER,
  protocolInfo  OCTET STRING
}
```

For example, register might be specified as the Group Secure Association Key Management Protocol (GSAKMP) [RFC4535] registration protocol. The OBJECT IDENTIFIER TBS would be followed by the parameters used in GSAKMP registration as specified in Appendix B.1.

3.3. Rekey Policy

The Rekey SA is defined in the MSEC Architecture. During the Rekey of a group, several changes can potentially be made:

- refresh/change group protection keys,

- update the policy token,

- change the group membership.

During Rekey, the membership of the group can be modified as well as refreshing the group traffic protection keys and updating the Policy Token.

This field is also specified as a sequence of protocols that will be used by the GC/KS.

3.4. Group Data Policy

The Data SA is the ultimate consumer of the group keys. The data field will indicate the keys and mechanisms that are to be used in communications between group members. There are several protocols that could make use of group keys, ranging from simple security applications that only need key for encryption and/or integrity protection to more complex configurable security protocols such as IPsec and Secure Real-time Transport Protocol (SRTP) [RFC3711]. The sequencing of the Data SA mechanisms are from "inside" to "outside". That is, the first Data SA defined in a policy token must act on the raw data. Any Data SA specified after that will be applied in turn.

```
DataProtocol ::= Protocol
```

4. Security Considerations

This document specifies the structure for a group policy token. As such, the structure as received by a group entity must be verifiably authentic. This policy token uses CMS to apply authentication through digital signatures. The security of this scheme relies upon a secure CMS implementation, choice of signature mechanism of appropriate strength for the group using the policy token, and secure, sufficiently strong keys. Additionally, it relies upon knowledge of a well-known Group Owner as the root of policy enforcement.

Furthermore, while the Group Owner may list alternate mechanisms for various functions, the group is only as strong as the weakest accepted mechanisms. As such, the Group Owner is responsible for providing only acceptable security mechanisms.

5. IANA Considerations

The following object identifiers have been assigned:

- id-ct-msec-token OBJECT IDENTIFIER ::= 1.3.6.1.5.5.12.1.1

- id-securitySuiteOne OBJECT IDENTIFIER ::= 1.3.6.1.5.5.12.2.1

- id-GSAKMPv1RegistrationProtocol
  OBJECT IDENTIFIER::= 1.3.6.1.5.5.12.3.1

- id-GSAKMPv1DeRegistrationProtocol
  OBJECT IDENTIFIER::= 1.3.6.1.5.5.12.3.2

- id-GSAKMPv1Rekey OBJECT IDENTIFIER::= 1.3.6.1.5.5.12.3.3

- id-rekeyNone OBJECT IDENTIFIER ::= 1.3.6.1.5.5.12.4.1

- id-rekeyMethodGSAKMPLKH
  OBJECT IDENTIFIER ::= 1.3.6.1.5.5.12.4.2

- id-reliabilityNone OBJECT IDENTIFIER ::= 1.3.6.1.5.5.12.5.1

- id-reliabilityResend OBJECT IDENTIFIER ::= 1.3.6.1.5.5.12.5.2

- id-reliabilityPost OBJECT IDENTIFIER ::= 1.3.6.1.5.5.12.5.3

- id-subGCKSSchemeNone OBJECT IDENTIFIER ::= 1.3.6.1.5.5.12.6.1

- id-subGCKSSchemeAutonomous
  OBJECT IDENTIFIER ::= 1.3.6.1.5.5.12.6.2

- id-genericDataSA OBJECT IDENTIFIER ::= 1.3.6.1.5.5.12.7.1

The Group Security Policy Token can be extended through specification. Extensions in the form of objects can be registered through IANA. Extensions requiring changes to the protocol structure will require an update to the tokenDefVersion field of the TokenID (see Section 3.1).

6. References

6.1. Normative References

[RFC4535] Harney, H., Meth, U., Colegrove, A., and G. Gross, "GSAKMP: Group Secure Association Key Management Protocol", RFC 4535, June 2006.

[RFC3280] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[RFC3852] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, July 2004.

[X.208] Recommendation X.208, Specification of Abstract Syntax Notation One (ASN.1), 1988.

[X.660] Recommendation X.660, Information Technology ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER), and Distinguished Encoding Rules (DER), 1997.

6.2. Informative References

[HCLM00] Harney, H., Colegrove, A., Lough, P., and U. Meth, "GSAKMP Token Specification", Work in Progress, February 2003.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC3740] Hardjono, T. and B. Weis, "The Multicast Group Security Architecture", RFC 3740, March 2004.

[HCM01] H. Harney, A. Colegrove, P. McDaniel, "Principles of Policy in Secure Groups", Proceedings of Network and Distributed Systems Security 2001 Internet Society, San Diego, CA, February 2001.

[HHMCD01] Hardjono, T., Harney, H., McDaniel, P., Colegrove, A., and P. Dinsmore, "Group Security Policy Token: Definition and Payloads", Work in Progress, August 2003.

7. Acknowledgements

The following individuals deserve recognition and thanks for their contributions, which have greatly improved this specification: Uri Meth, whose knowledge of GSAKMP and tokens was greatly appreciated as well as his help in getting this document submitted; Peter Lough, Thomas Hardjono, Patrick McDaniel, and Pete Dinsmore for their work on earlier versions of policy tokens; George Gross for the impetus to have a well-specified, extensible policy token; and Rod Fleischer for catching implementation issues.

The following technical works influenced the design of the Group Security Policy Token: [HCLM00], [HCM01], and [HHMCD01]

Appendix A. Core Policy Token ASN.1 Module

```
PolicyToken
    {1.3.6.1.5.5.12.0.1}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

Token ::= SEQUENCE {
  tokenInfo    TokenID,
  registration SEQUENCE OF Registration,
  rekey        SEQUENCE OF GroupMngmtProtocol,
  data         SEQUENCE OF DataProtocol
}

-------------------------------------------------------------
    -- Token ID

TokenID ::= SEQUENCE {
  tokenDefVersion INTEGER (1),     -- Group Security Policy Token v1
  groupName       OCTET STRING,
  edition         INTEGER OPTIONAL
}

LifeDate ::= CHOICE {
  gt       GeneralizedTime,
  utc      UTCTime,
  interval INTEGER
}

-------------------------------------------------------------
    -- Registration

Registration ::= SEQUENCE {
  register    GroupMngmtProtocol,
  de-register GroupMngmtProtocol
}

-------------------------------------------------------------
    -- GroupMngmtProtocol

GroupMngmtProtocol ::= CHOICE {
  none      NULL,
  supported Protocol
}
```

```
Protocol ::= SEQUENCE {
  protocol     OBJECT IDENTIFIER,
  protocolInfo OCTET STRING
}

-------------------------------------------------------------
    -- DataProtocol

DataProtocol ::= Protocol

-------------------------------------------------------------

END
```

Appendix B. GSAKMPv1 Base Policy

This appendix provides the data structures needed for when GSAKMP exchanges are used as the GroupMngmtProtocol for the registration, de-registration, and/or Rekey SAs. This GSAKMP Base Policy specification assumes familiarity with GSAKMP.

B.1. GSAKMPv1 Registration Policy

When GSAKMP is used in the Group Management Protocol for registration, the following object identifier is used in the core token.

```
id-GSAKMPv1RegistrationProtocol
                    OBJECT IDENTIFIER::= {1.3.6.1.5.5.12.3.1}
```

The registration policy for GSAKMP provides 1) information on authorizations for group roles, 2) access control information for group members, 3) the mechanisms used in the registration process, and 4) information on what transport the GSAKMP registration exchange will use.

```
GSAKMPv1RegistrationInfo ::= SEQUENCE {
  joinAuthorization JoinAuthorization,
  joinAccessControl SEQUENCE OF AccessControl,
  joinMechanisms    JoinMechanisms,
  transport         Transport
}
```

B.1.1. Authorization

joinAuthorization provides information on who is allowed to be a Group Controller Key Server (GC/KS) and a sub-GC/KS. It also can indicate if there are limitations on who can send data in a group.

```
JoinAuthorization ::= SEQUENCE {
  gCKS    GCKSName,
  subGCKS SEQUENCE OF GCKSName OPTIONAL,
  senders SenderAuthorization
}
```

The authorization information is in the form of an access control list indicating entity name and acceptable certification authority information for the entity’s certificate.

```
GCKSName ::= SEQUENCE OF UserCAPair
```

```
UserCAPair ::= SEQUENCE {
  groupEntity  GSAKMPID,
  cA           CertAuth
}
```

groupEntity is defined by type and value. The types are indicated by integers that correspond to the GSAKMP Identification types. When a portion of a defined name type is filled with an "*", this indicates a wildcard, representing any valid choice for a field. This allows the specification of an authorization rule that is a set of related names.

```
GSAKMPID ::= SEQUENCE {
  typeValue  INTEGER,
  typeData   OCTET STRING
}
```

The certificate authority is identified by the X.509 [RFC3280] key identifier.

```
CertAuth ::= KeyIdentifier
```

Senders within a group either can be all (indicating no sender restrictions) or can be an explicit list of those members authorized to send data.

```
SenderAuthorization ::= CHOICE {
  all     [0] NULL,
  limited [1] EXPLICIT SEQUENCE OF UserCAPair
}
```

B.1.2. AccessControl

joinAccessControl provides information on who is allowed to be a Group Member. The access control list is implemented as a set of permissions that the member must satisfy and a list of name rules and the certificate authority that each must satisfy. Additionally, a list of exclusions to the list may be provided.

```
AccessControl ::= SEQUENCE {
  permissions    [1] EXPLICIT SEQUENCE OF Permission OPTIONAL,
  accessRule     [2] EXPLICIT SEQUENCE OF UserCAPair,
  exclusionsRule [3] EXPLICIT SEQUENCE OF UserCAPair OPTIONAL
}
```

The permissions initially available are an abstract set of numeric levels that may be interpreted internal to a community.

```
Permission ::= CHOICE {
  simplePermission [1] SimplePermission
}

SimplePermission ::= ENUMERATED {
  one(1),
  two(2),
  three(3),
  four(4),
  five(5),
  six(6),
  seven(7),
  eight(8),
  nine(9)
}
```

B.1.3. JoinMechanisms

Allowable GSAKMP mechanism choices for a particular group are specified in joinMechanisms. Any set of JoinMechanism is acceptable from a policy perspective.

```
JoinMechanisms ::=  SEQUENCE OF JoinMechanism
```

Each set of mechanisms used in the GSAKMP Registration may be specified either as an explicitly defined set or as a pre-defined security suite.

```
JoinMechanism ::= CHOICE {
  alaCarte [0] Mechanisms,
  suite    [1] SecuritySuite
}
```

B.1.3.1. alaCarte

In an explicitly defined -- or alaCarte -- set, a mechanism is defined for the signature, the key exchange algorithm, the key wrapping algorithm, the type of acknowledgement data, and configuration data for the setting of timeouts.

```
Mechanisms ::=  SEQUENCE {
  signatureDef   SigDef,
  kEAlg          KEAlg,
  keyWrap        KeyWrap,
  ackData        AckData,
  opInfo         OpInfo
}
```

The signature definition requires specification of the signature algorithm for message signing. The INTEGER that defines the choice corresponds to the GSAKMP Signature type.

```
SigDef ::= SEQUENCE {
  sigAlgorithmID  INTEGER,
  hashAlgorithmID INTEGER
}
```

The INTEGER corresponding to hashAlgorithm will map to the GSAKMP Nonce Hash type values. This algorithm is used in computing the combined nonce.

The key exchange algorithm requires an integer to define the GSAKMP key creation type and may require additional per type data.

```
  KEAlg ::= SEQUENCE {
    keyExchangeAlgorithmID   INTEGER,
    keyExchangeAlgorithmData OCTET STRING OPTIONAL
  }
```

The keyWrap is the algorithm that is used to wrap the group key(s) and the policy token (if included). The integer corresponds to the GSAKMP encryption type.

```
  KeyWrap ::= INTEGER
```

Data may potentially be returned in a GSAKMP Key Download ACK/Failure message. The type of data required by a group is specified by AckData. No such field is currently supported or required.

```
  AckData ::= CHOICE {
    none [0] NULL
  }
```

OpInfo provides configuration data for the operation of GSAKMP registration. timeOut indicates the elapsed amount of time before a sent message is considered to be misrouted or lost. It is specified as the timestamp type LifeDate, previously defined in the core token. terse informs a GC/KS whether the group should be operated in terse (TRUE) or verbose (FALSE) mode. The optional timestamp field indicates whether a timestamp (TRUE) or a nonce (FALSE) is used for anti-replay protection. If the field is absent, the use of nonces is the default mode for GSAKMP registration.

```
OpInfo ::= SEQUENCE {
  timeOut  LifeDate,
  terse    BOOLEAN,
  timestamp BOOLEAN OPTIONAL
}
```

B.1.3.2. suite

If the choice of mechanism for the join is a predefined security suite, then it is identified by OBJECT IDENTIFIER (OID). Other security suites may be defined elsewhere by specification and registration of an OID.

```
SecuritySuite ::= OBJECT IDENTIFIER
```

The OID for security suite 1, as defined within the GSAKMPv1 specification, is

```
id-securitySuiteOne  OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.2.1}
```

B.1.4. Transport

transport indicates what protocol GSAKMP should ride over. The choice of udpRTJtcpOther indicates that the GSAKMP Request to Join message is carried by UDP and all other group establishment messages are carried by TCP.

```
Transport ::= CHOICE {
  tcp             [0] NULL,
  udp             [1] NULL,
  udpRTJtcpOther  [2] NULL
}
```

B.2. GSAKMPv1 Registration ASN.1 Module

```
GSAKMPv1RegistrationSA
    {1.3.6.1.5.5.12.0.2}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN
  EXPORTS
    GCKSName;

  IMPORTS
    LifeDate
      FROM PolicyToken {1.3.6.1.5.5.12.0.1}
```

```
      KeyIdentifier
        FROM PKIX1Implicit88 { iso(1) identified-organization(3)
          dod(6) internet(1) security(5) mechanisms(5) pkix(7)
          id-mod(0) id-pkix1-implicit(19) };

  id-GSAKMPv1RegistrationProtocol
                    OBJECT IDENTIFIER::= {1.3.6.1.5.5.12.7}

  GSAKMPv1RegistrationInfo ::= SEQUENCE {
    joinAuthorization JoinAuthorization,
    joinAccessControl SEQUENCE OF AccessControl,
    joinMechanisms    JoinMechanisms,
    transport         Transport
  }

  JoinAuthorization ::= SEQUENCE {
    gCKS    GCKSName,
    subGCKS SEQUENCE OF GCKSName OPTIONAL,
    senders SenderAuthorization
  }

  GCKSName ::= SEQUENCE OF UserCAPair

  UserCAPair ::= SEQUENCE {
    groupEntity GSAKMPID,
    cA          CertAuth
  }

  CertAuth ::= KeyIdentifier

  SenderAuthorization ::= CHOICE {
    all     [0] NULL,
    limited [1] EXPLICIT SEQUENCE OF UserCAPair
  }

  AccessControl ::= SEQUENCE {
    permissions    [1] EXPLICIT SEQUENCE OF Permission OPTIONAL,
    accessRule     [2] EXPLICIT SEQUENCE OF UserCAPair,
    exclusionsRule [3] EXPLICIT SEQUENCE OF UserCAPair OPTIONAL
  }

  Permission ::= CHOICE {
    simplePermission [1] SimplePermission
  }
```

```
SimplePermission ::= ENUMERATED {
  one(1),
  two(2),
  three(3),
  four(4),
  five(5),
  six(6),
  seven(7),
  eight(8),
  nine(9)
}

GSAKMPID ::= SEQUENCE {
  typeValue INTEGER,
  typeData  OCTET STRING
}

JoinMechanisms ::=  SEQUENCE OF JoinMechanism

JoinMechanism ::= CHOICE {
  alaCarte [0] Mechanisms,
  suite    [1] SecuritySuite
}

Mechanisms ::=  SEQUENCE {
  signatureDef SigDef,
  kEAlg        KEAlg,
  keyWrap      KeyWrap,
  ackData      AckData,
  opInfo       OpInfo
}

SecuritySuite ::= OBJECT IDENTIFIER

-- SECURITY SUITE ONE --
id-securitySuiteOne OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.2.1}

SigDef ::= SEQUENCE {
  sigAlgorithmID  INTEGER,
  hashAlgorithmID INTEGER
}

KEAlg ::= SEQUENCE {
  keyExchangeAlgorithmID   INTEGER,
  keyExchangeAlgorithmData OCTET STRING OPTIONAL
}

KeyWrap ::= INTEGER
```

```
AckData ::= CHOICE {
  none [0] NULL
}

OpInfo ::= SEQUENCE {
  timeOut   LifeDate,
  terse     BOOLEAN,
  timestamp BOOLEAN OPTIONAL
}

Transport ::= CHOICE {
  tcp            [0] NULL,
  udp            [1] NULL,
  udpRTJtcpOther [2] NULL
}

END
```

B.3. GSAKMPv1 De-Registration Policy

GSAKMP de-registration provides a method to notify a (S-)GC/KS that a member needs to leave a group. When GSAKMP is the de-registration Protocol for the Group, the following object identifier is used in the core token.

id-GSAKMPv1DeRegistrationProtocol OBJECT IDENTIFIER::= {1.3.6.1.5.5.12.3.2}

The de-registration policy provides the mechanisms needed for the de-registration exchange messages, an indication of whether the exchange is to be done using terse (TRUE) or verbose (FALSE) mode, and the transport used for the GSAKMP de-registration messages.

```
GSAKMPv1DeRegistrationInfo ::= SEQUENCE {
  leaveMechanisms  SEQUENCE OF LeaveMechanisms,
  terse            BOOLEAN,
  transport        Transport
}
```

The policy dictating the mechanisms needed for the de-registration exchange is defined by leaveMechanisms. This field is specified as

```
LeaveMechanisms ::= SEQUENCE {
  sigAlgorithm   INTEGER,
  hashAlgorithm  INTEGER,
  cA             KeyIdentifier
}
```

The INTEGER corresponding to sigAlgorithm will map to the GSAKMP Signature type values. This algorithm set is to be used for message signing.

The INTEGER corresponding to hashAlgorithm will map to the GSAKMP Nonce Hash type values. This algorithm is used in computing the combined nonce.

cA represents a trust point off of which the signer's certificate must certify. It is identified by the Public Key Infrastructure for X.509 Certificates (PKIX) KeyIdentifier [RFC3280] type.

transport will provide the expected transport for GSAKMP de-registration messages. Initially, either UDP or TCP will be the policy for a group.

```
  Transport ::= CHOICE {
    tcp [0] NULL,
    udp [1] NULL
  }
```

B.4. GSAKMPv1 De-Registration ASN.1 Module

```
GSAKMPv1DeRegistrationSA
    {1.3.6.1.5.5.12.0.3}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

  IMPORTS
    KeyIdentifier
      FROM PKIX1Implicit88 { iso(1) identified-organization(3)
        dod(6) internet(1) security(5) mechanisms(5) pkix(7)
        id-mod(0) id-pkix1-implicit(19) };

id-GSAKMPv1DeRegistrationProtocol
                OBJECT IDENTIFIER::= {1.3.6.1.5.5.12.3.2}

GSAKMPv1DeRegistrationInfo ::= SEQUENCE {
  leaveMechanisms SEQUENCE OF LeaveMechanisms,
  transport       Transport
}
```

```
LeaveMechanisms ::= SEQUENCE {
  sigAlgorithm  INTEGER,
  hashAlgorithm INTEGER,
  cA            KeyIdentifier
}

Transport ::= CHOICE {
  tcp [0] NULL,
  udp [1] NULL
}

END
```

B.5. GSAKMPv1 Rekey Policy

When GSAKMP is used as the Rekey Protocol for the Group, the following object identifier should be used in the core token as the rekey protocol:

```
id-GSAKMPv1Rekey     OBJECT IDENTIFIER::= {1.3.6.1.5.5.12.0.4}
```

The GSAKMP rekey policy provides authorization information, mechanisms for the GSAKMP rekey messages, indicators defining rekey event definitions that define when the GC/KS should send a rekey message, the protocol or method the rekey event will use, the rekey interval that will allow a member to recognize a failure in the rekey process, a reliability indicator that defines the method the rekey will use to increase the likelihood of a rekey delivery (if any), and finally an indication of how subordinate-GC/KSes will handle rekey. This policy also describes the specific rekey policy methods "None" and "GSAKMP LKH REKEY".

```
  GSAKMPv1RekeyInfo ::= SEQUENCE {
    authorization  RekeyAuthorization,
    mechanism      RekeyMechanisms,
    rekeyEventDef  RekeyEventDef,
    rekeyMethod    RekeyMethod,
    rekeyInterval  LifeDate,
    reliability    Reliability,
    subGCKSInfo    SubGCKSInfo
  }
```

B.5.1. Rekey Authorization

```
   RekeyAuthorization ::= GCKSName
```

B.5.2. Rekey Mechanisms

The policy dictating the mechanisms needed for rekey message processing is defined by RekeyMechanisms. This field is specified as

```
RekeyMechanisms ::= SEQUENCE {
  sigAlgorithm   INTEGER,
  hashAlgorithm  INTEGER
}
```

The INTEGER corresponding to sigAlgorithm will map to the GSAKMP Signature type values. This algorithm set is to be used for message signing.

The INTEGER corresponding to hashAlgorithm will map to the GSAKMP Nonce Hash type values. This algorithm is used in computing the combined nonce.

B.5.3. Rekey Event Definition

Rekey Event Definition provides information to the GC/KS about the system requirements for sending rekey messages. This allows definition of the rekey event in time as well as event-driven characteristics (a number of de-registration notifications as an example), or a combination of the two (e.g., after x de-registrations or 24 hours, whichever comes first).

```
RekeyEventDef ::= CHOICE {
  none         [0]  NULL,     -- never rekey
  timeOnly     [1]  LifeDate, -- rekey every x units
  event        [2]  INTEGER,  -- rekey after x events
  timeAndEvent [3]  TimeAndEvent
}
```

The LifeDate specifies the maximum time a group should exist between rekeys. This does not require clock synchronization as this is used with respect to a local clock (a GC/KS clock for sending rekey messages or a member clock for determining whether a message has been missed).

The INTEGER corresponding to the event is an indicator of the number of events a group should sustain before a rekey message is sent. This defines the events between rekeys. An example of a relevant event is de-registration notifications.

The TimeAndEvent is defined as a couple of the LifeDate and Integer policies.

```
TimeAndEvent ::= SEQUENCE {
  time   LifeDate, -- rekey after x units of time OR
  event  INTEGER   -- x events occur
}
```

B.5.4. Rekey Methods

The rekey method defines the policy of how the rekey is to be accomplished. This field is specified as

```
RekeyMethod ::= SEQUENCE {
  rekeyMethodType  OBJECT IDENTIFIER,
  rekeyMethodInfo  OCTET STRING
}
```

The rekeyMethodType will define the rekey method to be used by the group.

The rekeyMethodInfo will supply the GMs with the information they need to operate in the correct rekey mode.

B.5.4.1. Rekey Method NONE

The group defined to work without a rekey protocols supporting it is supported by the rekeyMethodType NONE. There is no RekeyMethodNoneInfo associated with this option.

```
id-rekeyNone OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.4.1}

RekeyMethodNoneInfo ::= NULL
```

B.5.4.2. Rekey Method GSAKMP LKH

The GSAKMP protocol specification defined an interpretation of the Logical Key Hierarchy (LKH) protocol as a rekey method. This method is supported by the following values.

```
id-rekeyMethodGSAKMPLKH OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.4.2}

RekeyMethodGSAKMPLKHInfo ::= INTEGER
```

The GSAKMP LKH method requires a gsakmp type value for identifying the cryptographic algorithm used to wrap the keys. This value maps to the GSAKMP encryption type.

B.5.5. Rekey Interval

Rekey interval defines the maximum delay the GM should see between valid rekeys. This provides a means to ensure the GM is synchronized, from a key management perspective, with the rest of the group. It is defined as a time/date stamp.

B.5.6. Rekey Reliability

The rekey message in the GSAKMP protocol is a single push message. There are reliability concerns with such non-acknowledged messages (i.e., message exchange). The Reliability policy defines the mechanism used to deal with these concerns.

```
Reliability ::= SEQUENCE {
  reliabilityMechanism   OBJECT IDENTIFIER,
  reliabilityMechContent OCTET STRING
}
```

The reliability mechanism is defined by an OBJECT IDENTIFIER and the information needed to operate that mechanism is defined as reliabilityMechContent and is an OCTET STRING (as before).

B.5.6.1. Rekey Reliability Mechanism None

In networks with adequate reliability, it may not be necessary to use a mechanism to improve reliability of the rekey message. For these networks the ReliabilityMechanism NONE is appropriate.

```
id-reliabilityNone OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.5.1}

ReliabilityContentNone ::= NULL
```

B.5.6.2. Rekey Reliability Mechanism Resend

In networks with unknown or questionable reliability, it may be necessary to use a mechanism to improve reliability of the Rekey Message. For these networks, the ReliabilityMechanism RESEND is potentially appropriate. This mechanism has the GC/KS repeatedly sending out the same message.

```
id-reliabilityResend OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.5.2}

ReliabilityResendInfo ::= INTEGER
```

The INTEGER value in the ReliabilityResendInfo indicates the number of times the message should be resent.

B.5.6.3. Rekey Reliability Mechanism Post

Another reliability mechanism is to post the rekey message on some service that will make it generally available. This is the reliabilityPost method.

```
id-reliabilityPost OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.5.3}

ReliabilityContentPost ::= IA5String
```

The IA5String associated with ReliabilityPost is the identifier of the posting site and rekey message.

B.5.7. Distributed Operation Policy

The policy dictating the relationships between GC/KS and S-GC/KS for distributed operations is defined as SubGCKSInfo. It is defined as a couple of a subGCKSScheme and some information relating to that Scheme in sGCKSContent.

```
SubGCKSInfo ::= SEQUENCE {
  subGCKSScheme OBJECT IDENTIFIER,
  sGCKSContent  OCTET STRING
}
```

B.5.7.1. No Distributed Operation

If the group is not to use S-GC/KS, then that Scheme would be SGCKSSchemeNone.

```
id-subGCKSSchemeNone OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.6.1}

SGCKSNoneContent ::= NULL
```

B.5.7.2. Autonomous Distributed Mode

If the group is to use S-GC/KS as defined in the GSAKMP specification as Autonomous mode, then that scheme would be SGCKSAutonomous.

```
id-subGCKSSchemeAutonomous
                      OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.6.2}

SGCKSAutonomous ::= SEQUENCE {
  authSubs  GCKSName,
  domain    OCTET STRING OPTIONAL
}
```

The policy information needed for autonomous mode is a list of authorized S-GC/KSes and restrictions on who they may serve. The domain field representing these restrictions is NULL for this version.

B.6. GSAKMPv1 Rekey Policy ASN.1 Module

```
GSAKMPv1RekeySA
     {1.3.6.1.5.5.12.0.4}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

  IMPORTS
    GCKSName
      FROM GSAKMPv1RegistrationSA  {1.3.6.1.5.5.12.0.2}
    LifeDate
      FROM PolicyToken  {1.3.6.1.5.5.12.0.1};

id-GSAKMPv1Rekey OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.0.4}

GSAKMPv1RekeyInfo ::= SEQUENCE {
  authorization RekeyAuthorization,
  mechanism     RekeyMechanisms,
  rekeyEventDef RekeyEventDef, -- tells the GCKS when to rekey
  rekeyMethod   RekeyMethod,
  rekeyInterval LifeDate,      -- member knows when to rejoin
  reliability   Reliability,   -- what mech will be used to
                               --   increase the likelihood
                               --   of rekey delivery
  subGCKSInfo   SubGCKSInfo    -- what subordinate GCKS needs
}

RekeyAuthorization ::= GCKSName

RekeyMechanisms ::= SEQUENCE {
  sigAlgorithm  INTEGER,
  hashAlgorithm INTEGER
}

RekeyEventDef ::= CHOICE {
  none         [0] NULL,              -- never rekey
  timeOnly     [1] EXPLICIT LifeDate, -- rekey every x units
  event        [2] INTEGER,           -- rekey after x events
  timeAndEvent [3] TimeAndEvent
}
```

```
TimeAndEvent ::= SEQUENCE {
  time  LifeDate, -- rekey after x units of time OR
  event INTEGER   -- x events occur
}

RekeyMethod ::= SEQUENCE {
  rekeyMethodType OBJECT IDENTIFIER,
  rekeyMethodInfo OCTET STRING
}

-- REKEY METHOD NONE --

id-rekeyNone OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.4.1}

RekeyMethodNoneInfo ::= NULL

-- REKEY METHOD GSAKMP LKH --

id-rekeyMethodGSAKMPLKH OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.4.2}

RekeyMethodGSAKMPLKHInfo ::= INTEGER -- gsakmp type value for
                                     --   wrapping mechanism

Reliability ::= SEQUENCE {
  reliabilityMechanism   OBJECT IDENTIFIER,
  reliabilityMechContent OCTET STRING
}

-- RELIABILITY MECHANISM NONE --

id-reliabilityNone OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.5.1}

ReliabilityContentNone ::= NULL

-- RELIABILITY MECHANISM RESEND --

id-reliabilityResend OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.5.2}

ReliabilityResendInfo ::= INTEGER -- # of times rekey message should
                                  --   be resent

-- RELIABILITY MECHANISM POST --

id-reliabilityPost OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.5.3}

ReliabilityContentPost ::= IA5String
```

```
SubGCKSInfo ::= SEQUENCE {
  subGCKSScheme OBJECT IDENTIFIER,
  sGCKSContent  OCTET STRING
}

id-subGCKSSchemeNone OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.6.1}

SGCKSNoneContent ::= NULL

id-subGCKSSchemeAutonomous OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.6.2}

SGCKSAutonomous ::= SEQUENCE {
  authSubs GCKSName,
  domain   OCTET STRING OPTIONAL
}

END
```

Appendix C. Data SA Policy

The Data SA provides the data structures needed for the protection of the data exchanged between group members. This appendix defines the data structures needed for a simple, generic security application making use of fixed security mechanisms. Such a Data SA requires only that keys delivered by the registration and rekey protocols be mapped to the service using them.

C.1. Generic Data Policy

The Generic Data Policy has the following identifier:

```
id-genericDataSA OBJECT IDENTIFIER :: = {1.3.6.1.5.5.12.7.1}
```

If an authentication mechanism is used within the security application, the key identifier (kMKeyID) used in the key management protocol is given, as well as an optional key expiration date. Likewise, if an encryption mechanism is used within the security application, the encryption key identifier is given, as well as an optional key expiration date (keyExpirationDate).

```
GenericDataSAInfo ::= SEQUENCE {
  authentication [0] EXPLICIT KeyInfo OPTIONAL,
  encryption     [1] EXPLICIT KeyInfo OPTIONAL
}

KeyInfo ::= SEQUENCE{
  kMKeyID           OCTET STRING,
  keyExpirationDate LifeDate OPTIONAL
}
```

C.2. Generic Data Policy ASN.1 Module

```
GenericDataSA
    {1.3.6.1.5.5.12.0.5}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- DATA APPLICATION:  Generic
-- This token specification is for data applications with
-- fixed security mechanisms.  Such data applications only
-- need a mapping of management protocol key identification
-- tags to security service.
```

```
     IMPORTS
       LifeDate
         FROM PolicyToken {1.3.6.1.5.5.12.0.1}

       KeyIdentifier
         FROM PKIX1Implicit88 { iso(1) identified-organization(3)
           dod(6) internet(1) security(5) mechanisms(5) pkix(7)
           id-mod(0) id-pkix1-implicit(19) };

   id-genericDataSA OBJECT IDENTIFIER ::= {1.3.6.1.5.5.12.7.1}

   GenericDataSAInfo ::= SEQUENCE {
     authentication [0] EXPLICIT KeyInfo OPTIONAL,
     encryption     [1] EXPLICIT KeyInfo OPTIONAL
   }

   KeyInfo ::= SEQUENCE{
     kMKeyID           OCTET STRING,
     keyExpirationDate LifeDate OPTIONAL
   }

   END
```

Authors' Addresses

Andrea Colegrove
SPARTA, Inc.
7110 Samuel Morse Drive
Columbia, MD 21046

Phone: (443) 430-8014
Fax: (443) 430-8163
EMail: acc@sparta.com

Hugh Harney
SPARTA, Inc.
7110 Samuel Morse Drive
Columbia, MD 21046

Phone: (443) 430-8032
Fax: (443) 430-8181
EMail: hh@sparta.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).