                        SNMP Administrative Model

Status of this Memo

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

Table of Contents

1. Abstract

   This memo presents an elaboration of the SNMP administrative model
   set forth in [1]. This model provides a unified conceptual basis for
   administering SNMP protocol entities to support

      o authentication and integrity,

      o privacy,

      o access control, and

      o the cooperation of multiple protocol entities.

   Please send comments to the SNMP Security Developers mailing list
   (snmp-sec-dev@tis.com).

2. Introduction

   This memo presents an elaboration of the SNMP administrative model
   set forth in [1]. It describes how the elaborated administrative
   model is applied to realize effective network management in a variety
   of configurations and environments.

   The model described here entails the use of distinct identities for
   peers that exchange SNMP messages. Thus, it represents a departure
   from the community-based administrative model set forth in [1]. By
   unambiguously identifying the source and intended recipient of each
   SNMP message, this new strategy improves upon the historical
   community scheme both by supporting a more convenient access control
   model and allowing for effective use of asymmetric (public key)
   security protocols in the future.

3. Elements of the Model

3.1 SNMP Party

   A SNMP party is a conceptual, virtual execution context whose
   operation is restricted (for security or other purposes) to an
   administratively defined subset of all possible operations of a
   particular SNMP protocol entity (see Section 3.2). Whenever a SNMP
   protocol entity processes a SNMP message, it does so by acting as a
   SNMP party and is thereby restricted to the set of operations defined

for that party. The set of possible operations specified for a SNMP
party may be overlapping or disjoint with respect to the sets of
other SNMP parties; it may also be a proper or improper subset of all
possible operations of the SNMP protocol entity.

Architecturally, each SNMP party comprises

   o a single, unique party identity,

   o a single authentication protocol and associated
     parameters by which all protocol messages originated by
     the party are authenticated as to origin and integrity,

   o a single privacy protocol and associated parameters by
     which all protocol messages received by the party are
     protected from disclosure,

   o a single MIB view (see Section 3.6) to which all
     management operations performed by the party are
     applied, and

   o a logical network location at which the party executes,
     characterized by a transport protocol domain and
     transport addressing information.

Conceptually, each SNMP party may be represented by an ASN.1 value
with the following syntax:

     SnmpParty ::= SEQUENCE {
       partyIdentity
           OBJECT IDENTIFIER,
       partyTDomain
           OBJECT IDENTIFIER,
       partyTAddr
           OCTET STRING,
       partyProxyFor
           OBJECT IDENTIFIER,
       partyMaxMessageSize
           INTEGER,
       partyAuthProtocol
           OBJECT IDENTIFIER,
       partyAuthClock
           INTEGER,
       partyAuthLastMsg
           INTEGER,
       partyAuthNonce
           INTEGER,

```
         partyAuthPrivate
             OCTET STRING,
         partyAuthPublic
             OCTET STRING,
         partyAuthLifetime
             INTEGER,
         partyPrivProtocol
             OBJECT IDENTIFIER,
         partyPrivPrivate
             OCTET STRING,
         partyPrivPublic
             OCTET STRING
       }
```

For each SnmpParty value that represents a SNMP party, the following
statements are true:

   o Its partyIdentity component is the party identity.

   o Its partyTDomain component is called the transport
     domain and indicates the kind of transport service by
     which the party receives network management traffic.
     An example of a transport domain is
     rfc1351Domain (SNMP over UDP, using SNMP
     parties).

   o Its partyTAddr component is called the transport
     addressing information and represents a transport
     service address by which the party receives network
     management traffic.

   o Its partyProxyFor component is called the proxied
     party and represents the identity of a second SNMP
     party or other management entity with which
     interaction may be necessary to satisfy received
     management requests. In this context, the value
     noProxy signifies that the party responds to received
     management requests by entirely local mechanisms.

   o Its partyMaxMessageSize component is called the
     maximum message size and represents the length in
     octets of the largest SNMP message this party is
     prepared to accept.

   o Its partyAuthProtocol component is called the
     authentication protocol and identifies a protocol and a
     mechanism by which all messages generated by the party

Davin, Galvin, & McCloghrie   [Page 4]

are authenticated as to integrity and origin. In this
context, the value noAuth signifies that messages
generated by the party are not authenticated as to
integrity and origin.

o Its partyAuthClock component is called the
authentication clock and represents a notion of the
current time that is specific to the party. The
significance of this component is specific to the
authentication protocol.

o Its partyAuthLastMsg component is called the
last-timestamp and represents a notion of time
associated with the most recent, authentic protocol
message generated by the party. The significance of this
component is specific to the authentication protocol.

o Its partyAuthNonce component is called the nonce
and represents a monotonically increasing integer
associated with the most recent, authentic protocol
message generated by the party. The significance of this
component is specific to the authentication protocol.

o Its partyAuthPrivate component is called the private
authentication key and represents any secret value
needed to support the authentication protocol. The
significance of this component is specific to the
authentication protocol.

o Its partyAuthPublic component is called the public
authentication key and represents any public value that
may be needed to support the authentication protocol.
The significance of this component is specific to the
authentication protocol.

o Its partyAuthLifetime component is called the
lifetime and represents an administrative upper bound
on acceptable delivery delay for protocol messages
generated by the party. The significance of this
component is specific to the authentication protocol.

o Its partyPrivProtocol component is called the privacy
protocol and identifies a protocol and a mechanism by
which all protocol messages received by the party are
protected from disclosure. In this context, the value
noPriv signifies that messages received by the party are
not protected from disclosure.

   o Its partyPrivPrivate component is called the private
     privacy key and represents any secret value needed to
     support the privacy protocol. The significance of this
     component is specific to the privacy protocol.

   o Its partyPrivPublic component is called the public
     privacy key and represents any public value that may be
     needed to support the privacy protocol. The significance
     of this component is specific to the privacy protocol.

   If, for all SNMP parties realized by a SNMP protocol entity, the
   authentication protocol is noAuth and the privacy protocol is noPriv,
   then that protocol entity is called non-secure.

3.2   SNMP Protocol Entity

   A SNMP protocol entity is an actual process which performs network
   management operations by generating and/or responding to SNMP
   protocol messages in the manner specified in [1]. When a protocol
   entity is acting as a particular SNMP party (see Section 3.1), the
   operation of that entity must be restricted to the subset of all
   possible operations that is administratively defined for that party.

   By definition, the operation of a SNMP protocol entity requires no
   concurrency between processing of any single protocol message (by a
   particular SNMP party) and processing of any other protocol message
   (by a potentially different SNMP party). Accordingly, implementation
   of a SNMP protocol entity to support more than one party need not be
   multi-threaded. However, there may be situations where implementors
   may choose to use multi-threading.

   Architecturally, every SNMP entity maintains a local database that
   represents all SNMP parties known to it -- those whose operation is
   realized locally, those whose operation is realized by proxy
   interactions with remote parties or devices, and those whose
   operation is realized by remote entities. In addition, every SNMP
   protocol entity maintains a local database that represents an access
   control policy (see Section 3.11) that defines the access privileges
   accorded to known SNMP parties.

3.3   SNMP Management Station

   A SNMP management station is the operational role assumed by a SNMP
   party when it initiates SNMP management operations by the generation
   of appropriate SNMP protocol messages or when it receives and
   processes trap notifications.

   Sometimes, the term SNMP management station is applied to partial

implementations of the SNMP (in graphics workstations, for example)
that focus upon this operational role. Such partial implementations
may provide for convenient, local invocation of management services,
but they may provide little or no support for performing SNMP
management operations on behalf of remote protocol users.

3.4   SNMP Agent

A SNMP agent is the operational role assumed by a SNMP party when it
performs SNMP management operations in response to received SNMP
protocol messages such as those generated by a SNMP management
station (see Section 3.3).

Sometimes, the term SNMP agent is applied to partial implementations
of the SNMP (in embedded systems, for example) that focus upon this
operational role. Such partial implementations provide for
realization of SNMP management operations on behalf of remote users
of management services, but they may provide little or no support for
local invocation of such services.

3.5   View Subtree

A view subtree is the set of all MIB object instances which have a
common ASN.1 OBJECT IDENTIFIER prefix to their names. A view subtree
is identified by the OBJECT IDENTIFIER value which is the longest
OBJECT IDENTIFIER prefix common to all (potential) MIB object
instances in that subtree.

3.6   MIB View

A MIB view is a subset of the set of all instances of all object
types defined according to the Internet-standard SMI [2] (i.e., of
the universal set of all instances of all MIB objects), subject to
the following constraints:

   o Each element of a MIB view is uniquely named by an
     ASN.1 OBJECT IDENTIFIER value. As such,
     identically named instances of a particular object type
     (e.g., in different agents) must be contained within
     different MIB views. That is, a particular object
     instance name resolves within a particular MIB view to
     at most one object instance.

   o Every MIB view is defined as a collection of view
     subtrees.

3.7   SNMP Management Communication

   A SNMP management communication is a communication from one specified
   SNMP party to a second specified SNMP party about management
   information that is represented in the MIB view of the appropriate
   party. In particular, a SNMP management communication may be

      o a query by the originating party about information in
        the MIB view of the addressed party (e.g., getRequest
        and getNextRequest),

      o an indicative assertion to the addressed party about
        information in the MIB view of the originating party
        (e.g., getResponse or trapNotification), or

      o an imperative assertion by the originating party about
        information in the MIB view of the addressed party
        (e.g., setRequest).

   A management communication is represented by an ASN.1 value with the
   syntax

```
      SnmpMgmtCom ::= [1] IMPLICIT SEQUENCE {
        dstParty
           OBJECT IDENTIFIER,
        srcParty
           OBJECT IDENTIFIER,
        pdu
           PDUs
      }
```

   For each SnmpMgmtCom value that represents a SNMP management
   communication, the following statements are true:

      o Its dstParty component is called the destination and
        identifies the SNMP party to which the communication
        is directed.

      o Its srcParty component is called the source and
        identifies the SNMP party from which the
        communication is originated.

      o Its pdu component has the form and significance
        attributed to it in [1].

3.8   SNMP Authenticated Management Communication

   A SNMP authenticated management communication is a SNMP management
   communication (see Section 3.7) for which the originating SNMP party
   is (possibly) reliably identified and for which the integrity of the
   transmission of the communication is (possibly) protected. An
   authenticated management communication is represented by an ASN.1
   value with the syntax

```
      SnmpAuthMsg ::= [1] IMPLICIT SEQUENCE {
        authInfo
           ANY, - defined by authentication protocol
        authData
           SnmpMgmtCom
      }
```

   For each SnmpAuthMsg value that represents a SNMP authenticated
   management communication, the following statements are true:

     o Its authInfo component is called the authentication
       information and represents information required in
       support of the authentication protocol used by the
       SNMP party originating the message. The detailed
       significance of the authentication information is specific
       to the authentication protocol in use; it has no effect on
       the application semantics of the communication other
       than its use by the authentication protocol in
       determining whether the communication is authentic or
       not.

     o Its authData component is called the authentication
       data and represents a SNMP management
       communication.

3.9   SNMP Private Management Communication

   A SNMP private management communication is a SNMP authenticated
   management communication (see Section 3.8) that is (possibly)
   protected from disclosure. A private management communication is
   represented by an ASN.1 value with the syntax

```
   SnmpPrivMsg ::= [1] IMPLICIT SEQUENCE {
     privDst
         OBJECT IDENTIFIER,
     privData
         [1] IMPLICIT OCTET STRING
   }
```

   For each SnmpPrivMsg value that represents a SNMP private management
   communication, the following statements are true:

     o Its privDst component is called the privacy destination
       and identifies the SNMP party to which the
       communication is directed.

     o Its privData component is called the privacy data and
       represents the (possibly encrypted) serialization
       (according to the conventions of [3] and [1]) of a SNMP
       authenticated management communication (see
       Section 3.8).

3.10   SNMP Management Communication Class

   A SNMP management communication class corresponds to a specific SNMP
   PDU type defined in [1]. A management communication class is
   represented by an ASN.1 INTEGER value according to the type of the
   identifying PDU (see Table 1).

```
                    Get            1
                    GetNext        2
                    GetResponse    4
                    Set            8
                    Trap          16
```

           Table 1: Management Communication Classes

   The value by which a communication class is represented is computed
   as 2 raised to the value of the ASN.1 context-specific tag for the
   appropriate SNMP PDU.

   A set of management communication classes is represented by the ASN.1
   INTEGER value that is the sum of the representations of the
   communication classes in that set. The null set is represented by the
   value zero.

3.11   SNMP Access Control Policy

   A SNMP access control policy is a specification of a local access
   policy in terms of the network management communication classes which
   are authorized between pairs of SNMP parties. Architecturally, such a
   specification comprises three parts:

      o the targets of SNMP access control - the SNMP parties
        that may perform management operations as requested
        by management communications received from other
        parties,

      o the subjects of SNMP access control - the SNMP parties
        that may request, by sending management
        communications to other parties, that management
        operations be performed, and

      o the policy that specifies the classes of SNMP
        management communications that a particular target is
        authorized to accept from a particular subject.

   Access to individual MIB object instances is determined implicitly
   since by definition each (target) SNMP party performs operations on
   exactly one MIB view. Thus, defining the permitted access of a
   (reliably) identified subject party to a particular target party
   effectively defines the access permitted by that subject to that
   target's MIB view and, accordingly, to particular MIB object
   instances.

   Conceptually, a SNMP access policy is represented by a collection of
   ASN.1 values with the following syntax:

      AclEntry ::= SEQUENCE {
        aclTarget
           OBJECT IDENTIFIER,
        aclSubject
           OBJECT IDENTIFIER,
        aclPrivileges
           INTEGER
      }

   For each such value that represents one part of a SNMP access policy,
   the following statements are true:

        o Its aclTarget component is called the target and
          identifies the SNMP party to which the partial policy
          permits access.

        o Its aclSubject component is called the subject and
          identifies the SNMP party to which the partial policy
          grants privileges.

        o Its aclPrivileges component is called the privileges and
          represents a set of SNMP management communication
          classes that are authorized to be processed by the
          specified target party when received from the specified
          subject party.

3.12   SNMP Proxy Party

   A SNMP proxy party is a SNMP party that performs management
   operations by communicating with another, logically remote party.

   When communication between a logically remote party and a SNMP proxy
   party is via the SNMP (over any transport protocol), then the proxy
   party is called a SNMP native proxy party. Deployment of SNMP native
   proxy parties is a means whereby the processing or bandwidth costs of
   management may be amortized or shifted -- thereby facilitating the
   construction of large management systems.

   When communication between a logically remote party and a SNMP proxy
   party is not via the SNMP, then the proxy party is called a SNMP
   foreign proxy party. Deployment of foreign proxy parties is a means
   whereby otherwise unmanageable devices or portions of an internet may
   be managed via the SNMP.

   The transparency principle that defines the behavior of a SNMP party
   in general applies in particular to a SNMP proxy party:

        The manner in which one SNMP party processes
        SNMP protocol messages received from another
        SNMP party is entirely transparent to the latter.

   The transparency principle derives directly from the historical SNMP
   philosophy of divorcing architecture from implementation. To this
   dichotomy are attributable many of the most valuable benefits in both
   the information and distribution models of the management framework,
   and it is the architectural cornerstone upon which large management
   systems may be built. Consistent with this philosophy, although the
   implementation of SNMP proxy agents in certain environments may
   resemble that of a transport-layer bridge, this particular
   implementation strategy (or any other!) does not merit special

recognition either in the SNMP management architecture or in standard
mechanisms for proxy administration.

Implicit in the transparency principle is the requirement that the
semantics of SNMP management operations are preserved between any two
SNMP peers. In particular, the "as if simultaneous" semantics of a
Set operation are extremely difficult to guarantee if its scope
extends to management information resident at multiple network
locations. For this reason, proxy configurations that admit Set
operations that apply to information at multiple locations are
discouraged, although such operations are not explicitly precluded by
the architecture in those rare cases where they might be supported in
a conformant way.

Also implicit in the transparency principle is the requirement that,
throughout its interaction with a proxy agent, a management station
is supplied with no information about the nature or progress of the
proxy mechanisms by which its requests are realized. That is, it
should seem to the management station -- except for any distinction
in underlying transport address -- as if it were interacting via SNMP
directly with the proxied device. Thus, a timeout in the
communication between a proxy agent and its proxied device should be
represented as a timeout in the communication between the management
station and the proxy agent. Similarly, an error response from a
proxied device should -- as much as possible -- be represented by the
corresponding error response in the interaction between the proxy
agent and management station.

3.13   Procedures

   This section describes the procedures followed by a SNMP protocol
   entity in processing SNMP messages. These procedures are independent
   of the particular authentication and privacy protocols that may be in
   use.

3.13.1   Generating a Request

   This section describes the procedure followed by a SNMP protocol
   entity whenever either a management request or a trap notification is
   to be transmitted by a SNMP party.

   1. An ASN.1 SnmpMgmtCom value is constructed for
      which the srcParty component identifies the originating
      party, for which the dstParty component identifies the
      receiving party, and for which the other component
      represents the desired management operation.

2. The local database is consulted to determine the
   authentication protocol and other relevant information
   for the originating SNMP party.

3. An ASN.1 SnmpAuthMsg value is constructed with
   the following properties:

   o Its authInfo component is constructed according
     to the authentication protocol specified for the
     originating party.

     In particular, if the authentication protocol for the
     originating SNMP party is identified as noAuth,
     then this component corresponds to the OCTET
     STRING value of zero length.

   o Its authData component is the constructed
     SnmpMgmtCom value.

4. The local database is consulted to determine the privacy
   protocol and other relevant information for the receiving
   SNMP party.

5. An ASN.1 SnmpPrivMsg value is constructed with the
   following properties:

   o Its privDst component identifies the receiving
     SNMP party.

   o Its privData component is the (possibly
     encrypted) serialization of the SnmpAuthMsg
     value according to the conventions of [3] and [1].

     In particular, if the privacy protocol for the
     receiving SNMP party is identified as noPriv, then
     the privData component is unencrypted.
     Otherwise, the privData component is processed
     according to the privacy protocol.

6. The constructed SnmpPrivMsg value is serialized
   according to the conventions of [3] and [1].

7. The serialized SnmpPrivMsg value is transmitted
   using the transport address and transport domain for
   the receiving SNMP party.

3.13.2   Processing a Received Communication

   This section describes the procedure followed by a SNMP protocol
   entity whenever a management communication is received.

   1. If the received message is not the serialization (according
      to the conventions of [3] and [1]) of an ASN.1
      SnmpPrivMsg value, then that message is discarded
      without further processing.

   2. The local database is consulted for information about
      the receiving SNMP party identified by the privDst
      component of the SnmpPrivMsg value.

   3. If information about the receiving SNMP party is absent
      from the local database, or specifies a transport domain
      and address which indicates that the receiving party's
      operation is not realized by the local SNMP protocol
      entity, then the received message is discarded without
      further processing.

   4. An ASN.1 OCTET STRING value is constructed
      (possibly by decryption, according to the privacy
      protocol in use) from the privData component of said
      SnmpPrivMsg value.

      In particular, if the privacy protocol recorded for the
      party is noPriv, then the OCTET STRING value
      corresponds exactly to the privData component of the
      SnmpPrivMsg value.

   5. If the OCTET STRING value is not the serialization
      (according to the conventions of [3] and [1]) of an ASN.1
      SnmpAuthMsg value, then the received message is
      discarded without further processing.

   6. If the dstParty component of the authData
      component of the obtained SnmpAuthMsg value is
      not the same as the privDst component of the
      SnmpPrivMsg value, then the received message is
      discarded without further processing.

   7. The local database is consulted for information about
      the originating SNMP party identified by the srcParty
      component of the authData component of the
      SnmpAuthMsg value.

8. If information about the originating SNMP party is
    absent from the local database, then the received
    message is discarded without further processing.

9. The obtained SnmpAuthMsg value is evaluated
    according to the authentication protocol and other
    relevant information associated with the originating
    SNMP party in the local database.

    In particular, if the authentication protocol is identified
    as noAuth, then the SnmpAuthMsg value is always
    evaluated as authentic.

10. If the SnmpAuthMsg value is evaluated as
    unauthentic, then the received message is discarded
    without further processing, and an authentication failure
    is noted.

11. The ASN.1 SnmpMgmtCom value is extracted from
    the authData component of the SnmpAuthMsg
    value.

12. The local database is consulted for access privileges
    permitted by the local access policy to the originating
    SNMP party with respect to the receiving SNMP party.

13. The management communication class is determined
    from the ASN.1 tag value associated with the
    SnmpMgmtCom value.

14. If the management communication class of the received
    message is either 16 or 4 (i.e., Trap or GetResponse) and
    this class is not among the access privileges, then the
    received message is discarded without further processing.

15. If the management communication class of the received
    message is not among the access privileges, then the
    received message is discarded without further processing
    after generation and transmission of a response message.
    This response message is directed to the originating
    SNMP party on behalf of the receiving SNMP party. Its
    var-bind-list and request-id components are identical
    to those of the received request. Its error-index
    component is zero and its error-status component is
    readOnly.

16. If the proxied party associated with the receiving SNMP
    party in the local database is identified as noProxy,

then the management operation represented by the
SnmpMgmtCom value is performed by the receiving
SNMP protocol entity with respect to the MIB view
identified with the receiving SNMP party according to
the procedures set forth in [1].

17. If the proxied party associated with the receiving SNMP
party in the local database is not identified as noProxy,
then the management operation represented by the
SnmpMgmtCom value is performed through
appropriate cooperation between the receiving SNMP
party and the identified proxied party.

In particular, if the transport domain associated with
the identified proxied party in the local database is
rfc1351Domain, then the operation requested by
the received message is performed by the generation of a
corresponding request to the proxied party on behalf of
the receiving party. If the received message requires a
response from the local SNMP protocol entity, then that
response is subsequently generated from the response (if
any) received from the proxied party corresponding to
the newly generated request.

3.13.3   Generating a Response

This section describes the procedure followed by a SNMP protocol
entity whenever a response to a management request is generated.

The procedure for generating a response to a SNMP management request
is identical to the procedure for transmitting a request (see Section
3.13.1), except for the derivation of the transport domain and
address information. In this case, the response is transmitted using
the transport domain and address from which the corresponding request
originated -- even if that is different from the transport
information recorded in the local database.

4. Application of the Model

This section describes how the administrative model set forth above
is applied to realize effective network management in a variety of
configurations and environments. Several types of administrative
configurations are identified, and an example of each is presented.

4.1   Non-Secure Minimal Agent Configuration

This section presents an example configuration for a minimal, non-
secure SNMP agent that interacts with one or more SNMP management

stations. Table 2 presents information about SNMP parties that is
known both to the minimal agent and to the manager, while Table 3
presents similarly common information about the local access policy.

As represented in Table 2, the example agent party operates at UDP
port 161 at IP address 1.2.3.4 using the party identity gracie; the
example manager operates at UDP port 2001 at IP address 1.2.3.5 using
the identity george. At minimum, a non-secure SNMP agent
implementation must provide for administrative configuration (and
non-volatile storage) of the identities and transport addresses of
two SNMP parties: itself and a remote peer. Strictly speaking, other
information about these two parties (including access policy
information) need not be configurable.

Suppose that the managing party george wishes to interrogate the
agent named gracie by issuing a SNMP GetNext request message. The
manager consults its local database of party information. Because the
authentication protocol for the party george is recorded as noAuth,
the GetNext request message generated by the manager is not

Identity	gracie (agent)	george (manager)
Domain	rfc1351Domain	rfc1351Domain
Address	1.2.3.4, 161	1.2.3.5, 2001
Proxied Party	noProxy	noProxy
Auth Prot	noAuth	noAuth
Auth Priv Key	""	""
Auth Pub Key	""	""
Auth Clock	0	0
Auth Last Msg	0	0
Auth Lifetime	0	0
Priv Prot	noPriv	noPriv
Priv Priv Key	""	""
Priv Pub Key	""	""

Table 2: Party Information for Minimal Agent

Target	Subject	Privileges
gracie	george	3
george	gracie	20

Table 3: Access Information for Minimal Agent

authenticated as to origin and integrity. Because, according to the
manager's database, the privacy protocol for the party gracie is
noPriv, the GetNext request message is not protected from disclosure.

Rather, it is simply assembled, serialized, and transmitted to the
transport address (IP address 1.2.3.4, UDP port 161) associated in
the manager's database with the party gracie.

When the GetNext request message is received at the agent, the
identity of the party to which it is directed (gracie) is extracted
from the message, and the receiving protocol entity consults its
local database of party information. Because the privacy protocol for
the party gracie is recorded as noPriv, the received message is
assumed not to be protected from disclosure. Similarly, the identity
of the originating party (george) is extracted, and the local party
database is consulted. Because the authentication protocol for the
party george is recorded as noAuth, the received message is
immediately accepted as authentic.

The received message is fully processed only if the access policy
database local to the agent authorizes GetNext request communications
by the party george with respect to the agent party gracie. The
access policy database presented as Table 3 authorizes such
communications (as well as Get operations).

When the received request is processed, a GetResponse message is
generated with gracie as the source party and george, the party from
which the request originated, as the destination party. Because the
authentication protocol for gracie is recorded in the local party
database as noAuth, the generated GetResponse message is not
authenticated as to origin or integrity. Because, according to the
local database, the privacy protocol for the party george is noPriv,
the response message is not protected from disclosure. The response
message is transmitted to the transport address from which the
corresponding request originated -- without regard for the transport
address associated with george in the local database.

When the generated response is received by the manager, the identity
of the party to which it is directed (george) is extracted from the
message, and the manager consults its local database of party
information. Because the privacy protocol for the party george is
recorded as noPriv, the received response is assumed not to be
protected from disclosure. Similarly, the identity of the originating
party (gracie) is extracted, and the local party database is
consulted. Because the authentication protocol for the party gracie
is recorded as noAuth, the received response is immediately accepted
as authentic.

The received message is fully processed only if the access policy
database local to the manager authorizes GetResponse communications
by the party gracie with respect to the manager party george. The
access policy database presented as Table 3 authorizes such response

messages (as well as Trap messages).

4.2   Secure Minimal Agent Configuration

This section presents an example configuration for a secure, minimal
SNMP agent that interacts with a single SNMP management station.
Table 4 presents information about SNMP parties that is known both to
the minimal agent and to the manager, while Table 5 presents
similarly common information about the local access policy.

The interaction of manager and agent in this configuration is very
similar to that sketched above for the non-secure minimal agent --
except that all protocol messages are authenticated as to origin and
integrity and protected from disclosure. This example requires
encryption in order to support distribution of secret keys via the
SNMP itself. A more elaborate example comprising an additional pair
of SNMP parties could support the exchange of non-secret information
in authenticated messages without incurring the cost of encryption.

An actual secure agent configuration may require SNMP parties for
which the authentication and privacy protocols are noAuth and noPriv,
respectively, in order to support clock synchronization (see [4]).
For clarity, these additional parties are not represented in this
example.

```
    Identity            ollie              stan
                        (agent)            (manager)
    Domain              rfc1351Domain      rfc1351Domain
    Address             1.2.3.4, 161       1.2.3.5, 2001
    Proxied Party       noProxy            noProxy
    Auth Prot           md5AuthProtocol    md5AuthProtocol
    Auth Priv Key       "0123456789ABCDEF" "GHIJKL0123456789"
    Auth Pub Key        ""                 ""
    Auth Clock          0                  0
    Auth Last Msg       0                  0
    Auth Lifetime       500                500
    Priv Prot           desPrivProtocol    desPrivProtocol
    Priv Priv Key       "MNOPQR0123456789" "STUVWX0123456789"
    Priv Pub Key        ""                 ""
```

   Table 4: Party Information for Secure Minimal Agent


```
        Target   Subject   Privileges
        ollie    stan      3
        stan     ollie     20
```

   Table 5: Access Information for Secure Minimal Agent

As represented in Table 4, the example agent party operates at UDP
port 161 at IP address 1.2.3.4 using the party identity ollie; the
example manager operates at UDP port 2001 at IP address 1.2.3.5 using
the identity stan. At minimum, a secure SNMP agent implementation
must provide for administrative configuration (and non-volatile
storage) of relevant information about two SNMP parties: itself and a
remote peer. Both ollie and stan authenticate all messages that they
generate by using the SNMP authentication protocol md5AuthProtocol
and their distinct, private authentication keys. Although these
private authentication key values ("0123456789ABCDEF" and
"GHIJKL0123456789") are presented here for expository purposes,
knowledge of private authentication keys is not normally afforded to
human beings and is confined to those portions of the protocol
implementation that require it.

When using the md5AuthProtocol, the public authentication key for
each SNMP party is never used in authentication and verification of
SNMP exchanges. Also, because the md5AuthProtocol is symmetric in
character, the private authentication key for each party must be
known to another SNMP party with which authenticated communication is
desired. In contrast, asymmetric (public key) authentication
protocols would not depend upon sharing of a private key for their
operation.

All protocol messages originated by the party stan are encrypted on
transmission using the desPrivProtocol privacy protocol and the
private key "STUVWX0123456789"; they are decrypted upon reception
according to the same protocol and key. Similarly, all messages
originated by the party ollie are encrypted on transmission using the
desPrivProtocol protocol and private privacy key "MNOPQR0123456789";
they are correspondingly decrypted on reception. As with
authentication keys, knowledge of private privacy keys is not
normally afforded to human beings and is confined to those portions
of the protocol implementation that require it.

4.3   Proxy Configuration

   This section presents examples of SNMP proxy configurations. On one
   hand, foreign proxy configurations provide the capability to manage
   non-SNMP devices. On the other hand, native proxy configurations
   allow an administrator to shift the computational burden of rich
   management functionality away from network devices whose primary task
   is not management. To the extent that SNMP proxy agents function as
   points of aggregation for management information, proxy
   configurations may also reduce the bandwidth requirements of large-
   scale management activities.

   The example configurations in this section are simplified for

clarity: actual configurations may require additional parties in
order to support clock synchronization and distribution of secrets.

4.3.1   Foreign Proxy Configuration

This section presents an example configuration by which a SNMP
management station may manage network elements that do not themselves
support the SNMP. This configuration centers on a SNMP proxy agent
that realizes SNMP management operations by interacting with a non-
SNMP device using a proprietary protocol.

Table 6 presents information about SNMP parties that is recorded in
the local database of the SNMP proxy agent. Table 7 presents
information about SNMP parties that is recorded in the local database
of the SNMP management station. Table 8 presents information about
the access policy specified by the local administration.

As represented in Table 6, the proxy agent party operates at UDP port
161 at IP address 1.2.3.5 using the party identity moe; the example
manager operates at UDP port 2002 at IP address 1.2.3.4 using the
identity larry. Both larry and moe authenticate all messages that
they generate by using the protocol md5AuthProtocol and their
distinct, private authentication keys. Although these private
authentication key values ("0123456789ABCDEF" and

Identity	larry	moe	curly
	(manager)	(proxy)	(proxied)
Domain	rfc1351Domain	rfc1351Domain	acmeMgmtPrtcl
Address	1.2.3.4, 2002	1.2.3.5, 161	0x98765432
Proxied Party	noProxy	curly	noProxy
Auth Prot	md5AuthProtocol	md5AuthProtocol	noAuth
Auth Priv Key	"0123456789ABCDEF"	"GHIJKL0123456789"	""
Auth Pub Key	""	""	""
Auth Clock	0	0	0
Auth Last Msg	0	0	0
Auth Lifetime	500	500	0
Priv Prot	noPriv	noPriv	noPriv
Priv Priv Key	""	""	""
Priv Pub Key	""	""	""

Table 6: Party Information for Proxy Agent

```
    Identity          larry               moe
                      (manager)           (proxy)
    Domain            rfc1351Domain       rfc1351Domain
    Address           1.2.3.4, 2002       1.2.3.5, 161
    Proxied Party     noProxy             noProxy
    Auth Prot         md5AuthProtocol     md5AuthProtocol
    Auth Priv Key     "0123456789ABCDEF"  "GHIJKL0123456789"
    Auth Pub Key      ""                  ""
    Auth Clock        0                   0
    Auth Last Msg     0                   0
    Auth Lifetime     500                 500
    Priv Prot         noPriv              noPriv
    Priv Priv Key     ""                  ""
    Priv Pub Key      ""                  ""
```

Table 7: Party Information for Management Station

```
        Target   Subject   Privileges
        moe      larry     3
        larry    moe       20
```

Table 8: Access Information for Foreign Proxy

"GHIJKL0123456789") are presented here for expository purposes,
knowledge of private keys is not normally afforded to human beings
and is confined to those portions of the protocol implementation that
require it.

Although all SNMP agents that use cryptographic keys in their
communication with other protocol entities will almost certainly
engage in private SNMP exchanges to distribute those keys, in order
to simplify this example, neither the management station nor the
proxy agent sends or receives private SNMP communications. Thus, the
privacy protocol for each of them is recorded as noPriv.

The party curly does not send or receive SNMP protocol messages;
rather, all communication with that party proceeds via a hypothetical
proprietary protocol identified by the value acmeMgmtPrtcl. Because
the party curly does not participate in the SNMP, many of the
attributes recorded for that party in a local database are ignored.

In order to interrogate the proprietary device associated with the
party curly, the management station larry constructs a SNMP GetNext
request and transmits it to the party moe operating (see Table 7) at
UDP port 161, and IP address 1.2.3.5. This request is authenticated
using the private authentication key "0123456789ABCDEF."

When that request is received by the party moe, the originator of the
message is verified as being the party larry by using local knowledge
(see Table 6) of the private authentication key "0123456789ABCDEF."
Because party larry is authorized to issue GetNext requests with
respect to party moe by the relevant access control policy (Table 8),
the request is accepted. Because the local database records the
proxied party for party moe as curly, the request is satisfied by its
translation into appropriate operations of the acmeMgmtPrtcl directed
at party curly. These new operations are transmitted to the party
curly at the address 0x98765432 in the acmeMgmtPrtcl domain.

When and if the proprietary protocol exchange between the proxy agent
and the proprietary device concludes, a SNMP GetResponse management
operation is constructed by the SNMP party moe to relay the results
to party larry. This response communication is authenticated as to
origin and integrity using the authentication protocol
md5AuthProtocol and private authentication key "GHIJKL0123456789"
specified for transmissions from party moe. It is then transmitted to
the SNMP party larry operating at the management station at IP
address 1.2.3.4 and UDP port 2002 (the source address for the
corresponding request).

When this response is received by the party larry, the originator of
the message is verified as being the party moe by using local
knowledge (see Table 7) of the private authentication key
"GHIJKL0123456789." Because party moe is authorized to issue
GetResponse communications with respect to party larry by the
relevant access control policy (Table 8), the response is accepted,
and the interrogation of the proprietary device is complete.

It is especially useful to observe that the database of SNMP parties
recorded at the proxy agent (Table 6) need be neither static nor
configured exclusively by the management station. For instance,
suppose that, in this example, the acmeMgmtPrtcl was a proprietary,
MAC-layer mechanism for managing stations attached to a local area
network. In such an environment, the SNMP party moe would reside at a
SNMP proxy agent attached to such a LAN and could, by participating
in the LAN protocols, detect the attachment and disconnection of
various stations on the LAN. In this scenario, the SNMP proxy agent
could easily adjust its local database of SNMP parties to support
indirect management of the LAN stations by the SNMP management
station. For each new LAN station detected, the SNMP proxy agent
would add to its database both an entry analogous to that for party
curly (representing the new LAN station itself) and an entry
analogous to that for party moe (representing a proxy for that new
station in the SNMP domain).

By using the SNMP to interrogate the database of parties held locally

by the SNMP proxy agent, a SNMP management station can discover and
interact with new stations as they are attached to the LAN.

4.3.2   Native Proxy Configuration

This section presents an example configuration that supports SNMP
native proxy operations -- indirect interaction between a SNMP agent
and a management station that is mediated by a second SNMP (proxy)
agent.

This example configuration is similar to that presented in the
discussion of SNMP foreign proxy above. In this example, however, the
party associated with the identity curly receives messages via the
SNMP, and, accordingly interacts with the SNMP proxy agent moe using
authenticated SNMP communications.

Table 9 presents information about SNMP parties that is recorded in
the local database of the SNMP proxy agent. Table 7 presents
information about SNMP parties that is recorded in the local database
of the SNMP management station. Table 10 presents information about
the access policy specified by the local administration.

As represented in Table 9, the proxy party operates at UDP port 161
at IP address 1.2.3.5 using the party identity moe;

Identity	larry (manager)	moe (proxy)	curly (proxied)
Domain	rfc1351Domain	rfc1351Domain	rfc1351Domain
Address	1.2.3.4, 2002	1.2.3.5, 161	1.2.3.6, 16
Proxied Party	noProxy	curly	noProxy
Auth Prot	md5AuthProtocol	md5AuthProtocol	md5AuthProtocol
Auth Priv Key	"0123456789ABCDEF"	"GHIJKL0123456789"	"MNOPQR0123456789"
Auth Pub Key	""	""	""
Auth Clock	0	0	0
Auth Last Msg	0	0	0
Auth Lifetime	500	500	500
Priv Prot	noPriv	noPriv	noPriv
Priv Priv Key	""	""	""
Priv Pub Key	""	""	""

Table 9: Party Information for Proxy Agent

```
          Target    Subject    Privileges
          moe       larry      3
          larry     moe        20
          curly     moe        3
          moe       curly      20
```

         Table 10: Access Information for Native Proxy

the example manager operates at UDP port 2002 at IP address 1.2.3.4
using the identity larry; the proxied party operates at UDP port 161
at IP address 1.2.3.6 using the party identity curly. Messages
generated by all three SNMP parties are authenticated as to origin
and integrity by using the authentication protocol md5AuthProtocol
and distinct, private authentication keys. Although these private key
values ("0123456789ABCDEF," "GHIJKL0123456789," and
"MNOPQR0123456789") are presented here for expository purposes,
knowledge of private keys is not normally afforded to human beings
and is confined to those portions of the protocol implementation that
require it.

In order to interrogate the proxied device associated with the party
curly, the management station larry constructs a SNMP GetNext request
and transmits it to the party moe operating (see Table 7) at UDP port
161 and IP address 1.2.3.5. This request is authenticated using the
private authentication key "0123456789ABCDEF."

When that request is received by the party moe, the originator of the
message is verified as being the party larry by using local knowledge
(see Table 9) of the private authentication key "0123456789ABCDEF."
Because party larry is authorized to issue GetNext (and Get) requests
with respect to party moe by the relevant access control policy
(Table 10), the request is accepted. Because the local database
records the proxied party for party moe as curly, the request is
satisfied by its translation into a corresponding SNMP GetNext
request directed from party moe to party curly. This new
communication is authenticated using the private authentication key
"GHIJKL0123456789" and transmitted to party curly at the IP address
1.2.3.6.

When this new request is received by the party curly, the originator
of the message is verified as being the party moe by using local
knowledge (see Table 9) of the private authentication key
"GHIJKL0123456789." Because party moe is authorized to issue GetNext
(and Get) requests with respect to party curly by the relevant access
control policy (Table 10), the request is accepted. Because the local
database records the proxied party for party curly as noProxy, the
GetNext request is satisfied by local mechanisms. A SNMP GetResponse
message representing the results of the query is then generated by

party curly. This response communication is authenticated as to
origin and integrity using the private authentication key
"MNOPQR0123456789" and transmitted to party moe at IP address 1.2.3.5
(the source address for the corresponding request).

When this response is received by party moe, the originator of the
message is verified as being the party curly by using local knowledge
(see Table 9) of the private authentication key "MNOPQR0123456789."
Because party curly is authorized to issue GetResponse communications
with respect to party moe by the relevant access control policy
(Table 10), the response is not rejected. Instead, it is translated
into a response to the original GetNext request from party larry.
This response is authenticated as to origin and integrity using the
private authentication key "GHIJKL0123456789" and is transmitted to
the party larry at IP address 1.2.3.4 (the source address for the
original request).

When this response is received by the party larry, the originator of
the message is verified as being the party moe by using local
knowledge (see Table 7) of the private authentication key
"GHIJKL0123456789." Because party moe is authorized to issue
GetResponse communications with respect to party larry by the
relevant access control policy (Table 10), the response is accepted,
and the interrogation is complete.

4.4   Public Key Configuration

This section presents an example configuration predicated upon a
hypothetical security protocol. This hypothetical protocol would be
based on asymmetric (public key) cryptography as a means for
providing data origin authentication (but not protection against
disclosure). This example illustrates the consistency of the
administrative model with public key technology, and the extension of
the example to support protection against disclosure should be
apparent.

```
   Identity            ollie                   stan
                       (agent)                 (manager)
   Domain              rfc1351Domain           rfc1351Domain
   Address             1.2.3.4, 161            1.2.3.5, 2004
   Proxied Party       noProxy                 noProxy
   Auth Prot           pkAuthProtocol          pkAuthProtocol
   Auth Priv Key       "0123456789ABCDEF"      ""
   Auth Pub Key        ""                      "ghijkl0123456789"
   Auth Clock          0                       0
   Auth Last Msg       0                       0
   Auth Lifetime       500                     500
   Priv Prot           noPriv                  noPriv
   Priv Priv Key       ""                      ""
   Priv Pub Key        ""                      ""
```

       Table 11: Party Information for Public Key Agent

The example configuration comprises a single SNMP agent that
interacts with a single SNMP management station. Tables 11 and 12
present information about SNMP parties that is by the agent and
manager, respectively, while Table 5 presents information about the
local access policy that is known to both manager and agent.

As represented in Table 11, the example agent party operates at UDP
port 161 at IP address 1.2.3.4 using the party identity ollie; the
example manager operates at UDP port 2004 at IP address 1.2.3.5 using
the identity stan. Both ollie and stan authenticate all messages that
they generate as to origin and integrity by using the hypothetical
SNMP authentication protocol pkAuthProtocol and their distinct,
private

```
    Identity            ollie                   stan
                        (agent)                 (manager)
    Domain              rfc1351Domain           rfc1351Domain
    Address             1.2.3.4, 161            1.2.3.5, 2004
    Proxied Party       noProxy                 noProxy
    Auth Prot           pkAuthProtocol          pkAuthProtocol
    Auth Priv Key       ""                      "GHIJKL0123456789"
    Auth Pub Key        "0123456789abcdef"      ""
    Auth Clock          0                       0
    Auth Last Msg       0                       0
    Auth Lifetime       500                     500
    Priv Prot           noPriv                  noPriv
    Priv Priv Key       ""                      ""
    Priv Pub Key        ""                      ""
```

   Table 12: Party Information for Public Key Management
             Station

authentication keys. Although these private authentication key values
("0123456789ABCDEF" and "GHIJKL0123456789") are presented here for
expository purposes, knowledge of private keys is not normally
afforded to human beings and is confined to those portions of the
protocol implementation that require it.

In most respects, the interaction between manager and agent in this
configuration is almost identical to that in the example of the
minimal, secure SNMP agent described above. The most significant
difference is that neither SNMP party in the public key configuration
has knowledge of the private key by which the other party
authenticates its transmissions. Instead, for each received
authenticated SNMP communication, the identity of the originator is
verified by applying an asymmetric cryptographic algorithm to the
received message together with the public authentication key for the
originating party. Thus, in this configuration, the agent knows the
manager's public key ("ghijkl0123456789") but not its private key
("GHIJKL0123456789"); similarly, the manager knows the agent's public
key ("0123456789abcdef") but not its private key
("0123456789ABCDEF").

For simplicity, privacy protocols are not addressed in this example
configuration, although their use would be necessary to the secure,
automated distribution of secret keys.

4.5   MIB View Configurations

This section describes a convention for the definition of MIB views
and, using that convention, presents example configurations of MIB
views for SNMP parties.

A MIB view is defined by a collection of view subtrees (see Section
3.6), and any MIB view may be represented in this way. Because MIB
view definitions may, in certain cases, comprise a very large number
of view subtrees, a convention for abbreviating MIB view definitions
is desirable.

The convention adopted in [5] supports abbreviation of MIB view
definitions in terms of families of view subtrees that are either
included in or excluded from the definition of the relevant MIB view.
By this convention, a table locally maintained by each SNMP entity
defines the MIB view associated with each SNMP party realized by that
entity. Each entry in the table represents a family of view subtrees
that (according to the status of that entry) is either included in or
excluded from the MIB view of some SNMP party. Each table entry
represents a subtree family as a pairing of an OBJECT IDENTIFIER
value (called the family name) together with a bitstring value
(called the family mask). The family mask indicates which

subidentifiers of the associated family name are significant to the
definition of the represented subtree family. For each possible MIB
object instance, that instance belongs to the view subtree family
represented by a particular table entry if

   o the OBJECT IDENTIFIER name of that MIB
     object instance comprises at least as many subidentifiers
     as does the family name for said table entry, and

   o each subidentifier in the name of said MIB object
     instance matches the corresponding subidentifier of the
     relevant family name whenever the corresponding bit of
     the associated family mask is non-zero.

The appearance of a MIB object instance in the MIB view for a
particular SNMP party is related to the membership of that instance
in the subtree families associated with that party in local table
entries:

   o If a MIB object instance belongs to none of the relevant
     subtree families, then that instance is not in the MIB
     view for the relevant SNMP party.

   o If a MIB object instance belongs to the subtree family
     represented by exactly one of the relevant table entries,
     then that instance is included in, or excluded from, the
     relevant MIB view according to the status of that entry.

   o If a MIB object instance belongs to the subtree families
     represented by more than one of the relevant table
     entries, then that instance is included in, or excluded
     from, the relevant MIB view according to the status of
     the single such table entry for which, first, the associated
     family name comprises the greatest number of
     subidentifiers, and, second, the associated family name is
     lexicographically greatest.

The subtree family represented by a table entry for which the
associated family mask is all ones corresponds to the single view
subtree identified by the family name for that entry. Because the
convention of [5] provides for implicit extension of family mask
values with ones, the subtree family represented by a table entry
with a family mask of zero length always corresponds to a single view
subtree.

```
   Party Identity  Status      Family Name    Family Mask
   lucy            include     internet       ""h
```

                Table 13: View Definition for Minimal Agent

Using this convention for abbreviating MIB view definitions, some of
the most common definitions of MIB views may be conveniently
expressed. For example, Table 13 illustrates the MIB view definitions
required for a minimal SNMP entity that locally realizes a single
SNMP party for which the associated MIB view embraces all instances
of all MIB objects defined within the internet network management
framework. The represented table has a single entry. The SNMP party
(lucy) for which that entry defines the MIB view is identified in the
first column. The status of that entry (include) signifies that any
MIB object instance belonging to the subtree family represented by
that entry may appear in the MIB view for party lucy. The family name
for that entry is internet, and the zero-length family mask value
signifies that the relevant subtree family corresponds to the single
view subtree rooted at that node.

Another example of MIB view definition (see Table 14) is that of a
SNMP protocol entity that locally realizes multiple SNMP parties with
distinct MIB views. The MIB view associated with the party lucy
comprises all instances of all MIB objects defined within the
internet network management framework, except those pertaining to the
administration of SNMP parties. In contrast, the MIB view attributed
to the party ricky contains only MIB object instances defined in the
system group of the internet-standard MIB together with those object
instances by which SNMP parties are administered.

A more complicated example of MIB view configuration illustrates the
abbreviation of related collections of view subtrees by view subtree
families (see Table 15). In this

```
   Party Identity  Status      Family Name    Family Mask
   lucy            include     internet       ""h
   lucy            exclude     snmpParties    ""h
   ricky           include     system         ""h
   ricky           include     snmpParties    ""h
```

              Table 14: View Definition for Multiple Parties

example, the MIB view associated with party lucy includes all object
instances in the system group of the internet-standard MIB together
with some information related to the second network interface
attached to the managed device. However, this interface-related
information does not include the speed of the interface. The family

mask value "FFA0"h in the second table entry signifies that a MIB
object instance belongs to the relevant subtree family if the initial
prefix of its name places it within the ifEntry portion of the
registration hierarchy and if the eleventh subidentifier of its name
is 2. The MIB object instance representing the speed of the second
network interface belongs to the subtree families represented by both
the second and third entries of the table, but that particular
instance is excluded from the MIB view for party lucy because the
lexicographically greater of the relevant family names appears in the
table entry with status exclude.

The MIB view for party ricky is also defined in this example. The
MIB view attributed to the party ricky includes all object instances
in the icmp group of the internet-standard MIB, together with all
information relevant to the fifth network interface attached to the
managed device. In addition, the MIB view attributed to party ricky
includes the number of octets received on the fourth attached network
interface.

While, as suggested by the examples above, a wide range of MIB view
configurations are efficiently supported by the abbreviated
representation of [5], prudent MIB design can sometimes further
reduce the size and complexity of the most

Party Identity	Status	Family Name	Family Mask
lucy	include	system	""h
lucy	include	{ ifEntry 0 2 }	"FFA0"h
lucy	exclude	{ ifSpeed 2 }	""h
ricky	include	icmp	""h
ricky	include	{ ifEntry 0 5 }	"FFA0"h
ricky	include	{ ifInOctets 4 }	""h

Table 15: More Elaborate View Definitions

likely MIB view definitions. On one hand, it is critical that
mechanisms for MIB view configuration impose no absolute constraints
either upon the access policies of local administrations or upon the
structure of MIB namespaces; on the other hand, where the most common
access policies are known, the configuration costs of realizing those
policies may be slightly reduced by assigning to distinct portions of
the registration hierarchy those MIB objects for which local policies
most frequently require distinct treatment. The relegation in [5] of
certain objects to a distinct arc in the MIB namespace is an example
of this kind of optimization.

5. Compatibility

   Ideally, all SNMP management stations and agents would communicate
   exclusively using the secure facilities described in this memo. In
   reality, many SNMP agents may implement only the insecure SNMP
   mechanisms described in [1] for some time to come.

   New SNMP agent implementations should never implement both the
   insecure mechanisms of [1] and the facilities described here. Rather,
   consistent with the SNMP philosophy, the burden of supporting both
   sorts of communication should fall entirely upon managers. Perhaps
   the best way to realize both old and new modes of communication is by
   the use of a SNMP proxy agent deployed locally on the same system
   with a management station implementation. The management station
   implementation itself operates exclusively by using the newer, secure
   modes of communication, and the local proxy agent translates the
   requests of the manager into older, insecure modes as needed.

   It should be noted that proxy agent implementations may require
   additional information beyond that described in this memo in order to
   accomplish the requisite translation tasks implicit in the definition
   of the proxy function. This information could easily be retrieved
   from a filestore.

6. Security Considerations

   It is important to note that, in the example configuration for native
   proxy operations presented in this memo, the use of symmetric
   cryptography does not securely prevent direct communication between
   the SNMP management station and the proxied SNMP agent.

   While secure isolation of the management station and the proxied
   agent can, according to the administrative model set forth in this
   memo, be realized using symmetric cryptography, the required
   configuration is more complex and is not described in this memo.
   Rather, it is recommended that native proxy configurations that
   require secure isolation of management station from proxied agent be
   implemented using security protocols based on asymmetric (or "public
   key") cryptography. However, no SNMP security protocols based on
   asymmetric cryptography are currently defined.

   In order to participate in the administrative model set forth in this
   memo, SNMP implementations must support local, non-volatile storage
   of the local party database. Accordingly, every attempt has been made
   to minimize the amount of non-volatile storage required.

7. References

     [1] Case, J., M. Fedor, M. Schoffstall, and J. Davin, The Simple
         Network Management Protocol", RFC 1157, University of Tennessee
         at Knoxville, Performance Systems International, Performance
         Systems International, and the MIT Laboratory for Computer
         Science, May 1990. (Obsoletes RFC 1098.)

     [2] Rose, M., and K. McCloghrie, "Structure and Identification of
         Management Information for TCP/IP based internets", RFC 1155,
         Performance Systems International, Hughes LAN Systems, May 1990.
         (Obsoletes RFC 1065.)

     [3] Information Processing -- Open Systems Interconnection --
         Specification of Basic Encoding Rules for Abstract Syntax
         Notation One (ASN.1), International Organization for
         Standardization/International Electrotechnical Institute, 1987,
         International Standard 8825.

     [4] Galvin, J., McCloghrie, K., and J. Davin, "SNMP Security
         Protocols", RFC 1352, Trusted Information Systems, Inc., Hughes
         LAN Systems, Inc., MIT Laboratory for Computer Science, July
         1992.

     [5] McCloghrie, K., Davin, J., and J. Galvin, "Definitions of Managed
         Objects for Administration of SNMP Parties", RFC 1353, Hughes LAN
         Systems, Inc., MIT Laboratory for Computer Science, Trusted
         Information Systems, Inc., July 1992.

8. Authors' Addresses

     James R. Davin
     MIT Laboratory for Computer Science
     545 Technology Square
     Cambridge, MA 02139

     Phone: (617) 253-6020
     EMail: jrd@ptt.lcs.mit.edu

     James M. Galvin
     Trusted Information Systems, Inc.
     3060 Washington Road, Route 97
     Glenwood, MD 21738

     Phone: (301) 854-6889
     EMail: galvin@tis.com

          Keith McCloghrie
          Hughes LAN Systems, Inc.
          1225 Charleston Road
          Mountain View, CA 94043

          Phone: (415) 966-7934
          EMail: kzm@hls.com