3.2 Secure Minimal Agent Configuration 23

3.3 Proxy Configuration 25

3.3.1 Foreign Proxy Configuration 26

3.3.2 Native Proxy Configuration 30

3.4 Public Key Configuration 33

4 Compatibility 35

5 Security Considerations 36

Contents

1 Introduction **1**

2 Elements of the Model **2**

2.1 SNMP Party 2

2.2 SNMP Protocol Entity 6

2.3 SNMP Management Station 6

2.4 SNMP Agent 7

2.5 View Subtree 7

2.6 MIB View 7

2.7 SNMP Management Communication 8

2.8 SNMP Authenticated Management Communication 9

2.9 SNMP Private Management Communication 10

2.10 SNMP Management Communication Class 11

2.11 SNMP Access Control Policy 12

2.12 SNMP Proxy Party 13

2.13 Procedures 15

2.13.1 Generating a Request 15

2.13.2 Processing a Received Communication 16

2.13.3 Generating a Response 19

3 Application of the Model **20**

3.1 Non-Secure Minimal Agent Configuration 20

References

[1] Jeffrey D. Case, Mark S. Fedor, Martin L. Schoffstall, and James R. Davin. *A Simple Network Management Protocol.* Request for Comments 1157, DDN Network Information Center, SRI International, May 1990.

[2] Marshall T. Rose and Keith McCloghrie. *Structure and Identification of Management Information for TCP/IP based internets.* Request for Comments 1155, DDN Network Information Center, SRI International, May 1990.

[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.

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.

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.

5 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.

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 **pkAuthProt** and their distinct, private authentication keys. Although these private authentication key values ("ABCD" and "EFGH") 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 as to their origin and integrity. 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 ("efgh") but not its private key ("EFGH"); similarly, the manager knows the agent's public key ("abcd") but not its private key ("ABCD").

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 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

Identity	**ollie** (agent)	**stan** (manager)
Domain	**rfc1157**	**rfc1157**
Address	1.2.3.4, 161	1.2.3.5, 2004
Proxied Party	**noProxy**	**noProxy**
Auth Prot	**pkAuthProt**	**pkAuthProt**
Auth Priv Key	"ABCD"	""
Auth Pub Key	""	"efgh"
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

Identity	**ollie** (agent)	**stan** (manager)
Domain	**rfc1157**	**rfc1157**
Address	1.2.3.4, 161	1.2.3.5, 2004
Proxied Party	**noProxy**	**noProxy**
Auth Prot	**pkAuthProt**	**pkAuthProt**
Auth Priv Key	""	"EFGH"
Auth Pub Key	"abcd"	""
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

local knowledge (see Table 9) of the private authentication key "JKLM." Because party **curly** is authorized to issue Get responses with respect to party **moe** by the relevant access control policy (Table 10), the response is not rejected. Instead, it is translated into a GetNext response that corresponds to the original GetNext request from party **larry**. This response is authenticated as to origin and integrity using the private authentication key "EFGH" and is transmitted to the party **larry** at IP address 1.2.3.5 (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 "EFGH." Because party **moe** is authorized to issue Get responses with respect to party **larry** by the relevant access control policy (Table 10), the response is accepted, and the interrogation is complete.

3.4 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.

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

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 "ABCD."

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 "ABCD." 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 "EFGH" 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 "EFGH." 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 Get response 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 "JKLM" 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

Target	Subject	Privileges
moe	**larry**	3
larry	**moe**	20

Table 8: Access Information for Foreign Proxy

Identity	**larry** (manager)	**moe** (proxy)	**curly** (proxied)
Domain	**rfc1157**	**rfc1157**	**rfc1157**
Address	1.2.3.4, 2003	1.2.3.5, 161	1.2.3.6, 161
Proxied Party	**noProxy**	**curly**	**noProxy**
Auth Prot	**md4AuthProt**	**md4AuthProt**	**md4AuthProt**
Auth Priv Key	"ABCD"	"EFGH"	"JKLM"
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

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.

3.3.2 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 highly similar to that presented in the discussion of SNMP foreign proxy in the previous section. 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 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 2003 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**. All three SNMP parties authenticate all messages that they generate as to origin and integrity by using the authentication protocol **md4AuthProt** and their distinct, private authentication keys. Although these private key values ("ABCD," "EFGH," and "JKLM") are presented here for expository purposes, knowledge of private keys is not

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 represent the results. This response communication is authenticated as to origin and integrity using the authentication protocol **md4AuthProt** and private authentication key "EFGH" specified for transmissions from party **moe**. It is then transmitted 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 "EFGH." Because party **moe** is authorized to issue Get responses 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

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. Although the administrative model set forth here can easily accommodate asymmetric protocols, none are currently defined for use with the SNMP.

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 "ABCD."

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 "ABCD." 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

Identity	**larry** (manager)	**moe** (proxy)	**curly** (proxied)
Domain	**rfc1157**	**rfc1157**	**acmeMgmtPrtcl**
Address	1.2.3.4, 2002	1.2.3.5, 161	0x98765432
Proxied Party	**noProxy**	**curly**	**noProxy**
Auth Prot	**md4AuthProt**	**md4AuthProt**	**noAuth**
Auth Priv Key	"ABCD"	"EFGH"	""
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** (manager)	**moe** (proxy)
Domain	**rfc1157**	**rfc1157**
Address	1.2.3.4, 2002	1.2.3.5, 161
Proxied Party	**noProxy**	**noProxy**
Auth Prot	**md4AuthProt**	**md4AuthProt**
Auth Priv Key	"ABCD"	"EFGH"
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

whose primary task is not management, via a native proxy configuration. Among a variety of other benefits, to the extent that SNMP proxy agents function as points of aggregation for management information, configurations of this sort may also reduce the bandwidth requirements of large-scale management activities.

3.3.1 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 performs SNMP management operations by communicating 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 **md4AuthProt** and their distinct, private authentication keys. Although these private authentication key values ("ABCD" and "EFGH") 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.

Using the **md4AuthProt**, the public authentication key for each SNMP party is irrelevant. Also, because the **md4AuthProt** is symmetric in character, the private

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 **md4AuthProt** and their distinct, private authentication keys. Although these private authentication key values ("ABCD" and "EFGH") 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.

Using the **md4AuthProt**, the public authentication key for each SNMP party is irrelevant. Also, because the **md4AuthProt** 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. Although the administrative model set forth here can easily accommodate asymmetric protocols, none are currently defined for use with the SNMP.

All protocol messages originated by the party **stan** are encrypted on transmission using the **desPrivProt** privacy protocol and the private key "QRST"; 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 **desPrivProt** protocol and private authentication key "JKLM"; they are correspondingly decrypted on reception.

3.3 Proxy Configuration

This section presents example configurations for SNMP proxy management. On the one hand, proxy management via a so-called foreign proxy configuration provides the capability to manage non-SNMP devices; on the other, it allows an administrator to shift the computational burden of rich management functionality away from network devices

Identity	**ollie** (agent)	**stan** (manager)
Domain	**rfc1157**	**rfc1157**
Address	1.2.3.4, 161	1.2.3.5, 2001
Proxied Party	**noProxy**	**noProxy**
Auth Prot	**md4AuthProt**	**md4AuthProt**
Auth Priv Key	"ABCD"	"EFGH"
Auth Pub Key	""	""
Auth Clock	0	0
Auth Last Msg	0	0
Auth Lifetime	500	500
Priv Prot	**desPrivProt**	**desPrivProt**
Priv Priv Key	"JKLM"	"QRST"
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

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 agent 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).

3.2 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.

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

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 to not 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 GetNext operations).

When the received request is processed, a GetResponse message is generated by the agent **gracie** for the originating peer **george**. Because the authentication protocol for **gracie** is recorded in the local party database as **noAuth**, the GetResponse message generated by the manager 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. Rather, as required by [1], the response message is directed 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

Identity	**gracie** (agent)	**george** (manager)
Domain	**rfc1157**	**rfc1157**
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

3 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 examples are given of how each may be realized.

3.1 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 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

receiving SNMP protocol entity with respect to the MIB view identified with the receiving SNMP party identity according to the procedures set forth in [1].

16. If the proxied party associated with the receiving SNMP party in the local database is not identified as **noProxy**, then the SNMP management communication represented by the **SnmpMgmtCom** value is performed by 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 **rfc1157Domain**, 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.

2.13.3 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 2.13.1), except for the derivation of the transport domain and address information. In this case, the response is transmitted using the transport domain and address (not from the local database but) from which the corresponding request originated.

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 is not among the access privileges, then the received message is discarded after first checking if the class is not 16 (i.e., not a trap), and if so, generating a response to the originating SNMP party on behalf of the receiving SNMP party for which the **var-bind-list** and **request-id** components are identical to those of the received request, and for which the value of the **error-index** component is zero, and for which the value of the **error-status** component is **readOnly**.

15. If the proxied party associated with the receiving SNMP party in the local database is identified as **noProxy**, then the SNMP management communication represented by the **SnmpMgmtCom** value is performed by the

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 performed 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) from the **privData** component of said **SnmpPrivMsg** value according to the privacy protocol in use.

   In particular, if the privacy protocol recorded for the party is identified as **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.

- 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.
- 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:

   - Its **privDst** component identifies the receiving SNMP party.
   - 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.

2.13.2 Processing a Received Communication

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

That is, it should seem — except for any distinction in an underlying transport address — to the management station as if it were interacting 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.

2.13 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.

2.13.1 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 specifies 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:

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 to a SNMP proxy party. In particular:

> 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 the interaction between a management station and a proxy agent not supply information to the station about the nature or progress of the mechanisms by which its requests are realized.

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

- Its **aclTarget** component is called the *target* and identifies the SNMP party to which the partial policy permits access.
- Its **aclSubject** component is called the *subject* and identifies the SNMP party to which the partial policy grants privileges.
- 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.

2.12 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*. There are multiple reasons why a SNMP native proxy party might be used. For example, 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

2.11 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 particular SNMP parties. Architecturally, such a specification comprises three parts:

- the *targets* of SNMP access control – the SNMP parties that may perform management operations as requested by management communications received from other parties,
- 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
- the *policy* that specifies the SNMP management communications classes that a particular subject is authorized to request of a particular target.

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
}
```

- Its **privDst** component is called the *private destination* and identifies the SNMP party to which the communication is directed.

- Its **privData** component is called the *private data* and represents the (possibly encrypted) serialization (according to the conventions of [3] and [1]) of a SNMP authenticated management communication (see section 2.8).

2.10 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).

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.

Get	1
GetNext	2
GetResponse	4
Set	8
Trap	16

Table 1: Management Communication Classes

```
}
```

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

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

2.9 SNMP Private Management Communication

A *SNMP private management communication* is a SNMP authenticated management communication (see section 2.8) 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:

```
        srcParty
            OBJECT IDENTIFIER,
        pdu
            PDUs
    }
```

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

- Its **dstParty** component is called the *destination* and identifies the SNMP party to which the communication is directed.
- Its **srcParty** component is called the *source* and identifies the SNMP party from which the communication is originated.
- Its **pdu** component has the form and significance attributed to it in [1].

2.8 SNMP Authenticated Management Communication

A *SNMP authenticated management communication* is a SNMP management communication (see section 2.7) 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
```

- 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.

- Every MIB view is defined as a collection of view subtrees that are mutually disjoint.

2.7 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:

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

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

- 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,
```

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.

2.4 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 2.3).

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.

2.5 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.

2.6 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:

2.2 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 2.1), 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 protocol entity maintains a local database that represents all SNMP parties known to it — those whose operation is performed locally, those whose operation is performed through communication with remote proxied parties, and those whose operation is performed by remote parties. In addition, every SNMP protocol entity maintains a local database that represents an access control policy (see section 2.11) that defines the access privileges associated with known SNMP parties.

2.3 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,

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

- 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.

- 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.

- Its **partyPrivProt** component is called the *privacy protocol* and identifies a protocol 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.

- Its **partyPrivPrivate** component is called the *private privacy key* and represents any secret value that may be needed to support the privacy protocol.

- Its **partyPrivPublic** component is called the *public privacy key* and represents any public value that may be needed to support 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*.

domain and indicates the kind of transport service by which the party receives network management traffic. An example of a transport domain is **rfc1157Domain** (SNMP over UDP).

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

- 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.

- Its **partyAuthProt** component is called the *authentication protocol* and identifies a protocol by which all messages generated by the party are authenticated as to origin and integrity. In this context, the value **noAuth** signifies that messages generated by the party are not authenticated as to origin and integrity.

- 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.

- Its **partyAuthLastMsg** component is called the *ratchet* 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.

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

```
SnmpParty ::= SEQUENCE {
    partyIdentity
        OBJECT IDENTIFIER,
    partyTDomain
        OBJECT IDENTIFIER,
    partyTAddr
        OCTET STRING,
    partyProxyFor
        OBJECT IDENTIFIER,
    partyAuthProt
        OBJECT IDENTIFIER,
    partyAuthClock
        TimeTicks,
    partyAuthLastMsg
        TimeTicks,
    partyAuthNonce
        INTEGER,
    partyAuthPrivate
        OCTET STRING,
    partyAuthPublic
        OCTET STRING,
    partyAuthLifetime
        INTEGER,
    partyPrivProt
        OBJECT IDENTIFIER,
    partyPrivPrivate
        OCTET STRING,
    partyPrivPublic
        OCTET STRING
}
```

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

- Its **partyIdentity** component is called the *identity* of the party and corresponds to the party identity.
- Its **partyTDomain** component is called the *transport*

2 Elements of the Model

2.1 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 2.2). 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

- a single, unique party identity,
- a single authentication protocol and associated parameters by which all protocol messages originated by the party are authenticated as to origin and integrity,
- a single privacy protocol and associated secret by which all protocol messages received by the party are protected from disclosure,
- a single MIB view (see section 2.6) to which all management operations performed by the party are applied, and
- a logical network location at which the party executes, characterized by a transport protocol domain and transport addressing information.

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

SNMP Administrative Model

DRAFT

April 8, 1991

1 Introduction

This paper 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

- authentication and integrity,
- privacy,
- access control, and
- the cooperation of multiple protocol entities.

This paper also 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.