Security Protocols for Version 2 of the Simple Network Management Protocol (SNMPv2)

19 March 1995

draft-ietf-snmpv2-sec-ds-01.txt

Jeffrey D. Case
SNMP Research, Inc.
case@snmp.com

James Galvin
Trusted Information Systems
galvin@tis.com

Keith McCloghrie
Cisco Systems, Inc.
kzm@cisco.com

Marshall T. Rose
Dover Beach Consulting, Inc.
mrose@dbc.mtview.ca.us

Steven Waldbusser
Carnegie Mellon University
waldbusser@cmu.edu

Status of this Memo

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet- Drafts as reference material or to cite them other than as ``work in progress.''

To learn the current status of any Internet-Draft, please check the ``1id-abstracts.txt'' listing contained in the Internet- Drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

1. Introduction

A management system contains: several (potentially many) nodes, each with a processing entity, termed an agent, which has access to management instrumentation; at least one management station; and, a management protocol, used to convey management information between the agents and management stations. Operations of the protocol are carried out under an administrative framework which defines authentication, authorization, access control, and privacy policies.

Management stations execute management applications which monitor and control managed elements. Managed elements are devices such as hosts, routers, terminal servers, etc., which are monitored and controlled via access to their management information.

In the Administrative Infrastructure for SNMPv2 document [1], each SNMPv2 party is, by definition, associated with a single authentication protocol and a single privacy protocol. It is the purpose of this document, Security Protocols for SNMPv2, to define one such authentication and one such privacy protocol.

The authentication protocol provides a mechanism by which SNMPv2 messages transmitted by a party may be reliably identified as having originated from that party. The authentication protocol defined in this memo also reliably determines that the message received is the message that was sent.

The privacy protocol provides a mechanism by which SNMPv2 messages transmitted to a party are protected from disclosure. The privacy protocol in this memo specifies that only authenticated messages may be protected from disclosure.

These protocols are secure alternatives to the so-called "noAuth/noPriv" protocol defined in [1].

> USE OF THE noAuth/noPriv PROTOCOL ALONE DOES NOT CONSTITUTE SECURE NETWORK MANAGEMENT. THEREFORE, A NETWORK MANAGEMENT SYSTEM THAT IMPLEMENTS ONLY THE noAuth/noPriv PROTOCOL IS NOT CONFORMANT TO THIS SPECIFICATION.

The Digest Authentication Protocol is described in Section 3. It provides a data integrity service by transmitting a message digest - computed by the originator and verified by the recipient - with each SNMPv2 message. The data origin authentication service is provided by prefixing the message with a secret value known only to the originator

and recipient, prior to computing the digest. Thus, data integrity is supported explicitly while data origin authentication is supported implicitly in the verification of the digest.

The Symmetric Privacy Protocol is described in Section 4. It protects messages from disclosure by encrypting their contents according to a secret cryptographic key known only to the originator and recipient. The additional functionality provided by this protocol is assumed to justify its additional computational cost.

The Digest Authentication Protocol depends on the existence of loosely synchronized clocks between the originator and recipient of a message. The protocol specification makes no assumptions about the strategy by which such clocks are synchronized. Section 5.3 presents one strategy that is particularly suited to the demands of SNMP network management.

Both protocols described here require the sharing of secret information between the originator of a message and its recipient. The protocol specifications assume the existence of the necessary secrets. The selection of such secrets and their secure distribution to appropriate parties may be accomplished by a variety of strategies. Section 5.4 presents one such strategy that is particularly suited to the demands of SNMP network management.

1.1. A Note on Terminology

For the purpose of exposition, the original Internet-standard Network Management Framework, as described in RFCs 1155, 1157, and 1212, is termed the SNMP version 1 framework (SNMPv1). The current framework is termed the SNMP version 2 framework (SNMPv2).

1.1.1. Change Log

For the 19 March version: +

- The changes adopted by the SNMPv2 Working Group. +

For the 1 November version:

- recast RFC 1446 into an Internet-Draft,
- fixed typos,

- rewrote various paragraphs, sentences, phrases throughout the document to use less formal, but more easily understood, language, and to omit unnecessary/redundant text.

- rewrote text in various sections on the maintenance of party secrets to be consistent with the use of the desPrivProtocol being optional.

- removed text describing implementation-specific considerations.

- incorporated new text on the creation of parties, discussing "cloning" as defined in the Party MIB.

1.2. Threats

Several of the classical threats to network protocols are applicable to the network management problem and therefore would be applicable to any SNMPv2 security protocol. Other threats are not applicable to the network management problem. This section discusses principal threats, secondary threats, and threats which are of lesser importance.

The principal threats against which any SNMPv2 security protocol should provide protection are:

Modification of Information
: The SNMPv2 protocol provides the means for management stations to interrogate and to manipulate the value of objects in a managed agent. The modification threat is the danger that some party may alter in-transit messages generated by an authorized party in such a way as to effect unauthorized management operations, including falsifying the value of an object.

Masquerade
: The SNMPv2 administrative model includes an access control model. Access control necessarily depends on knowledge of the origin of a message. The masquerade threat is the danger that management operations not authorized for some party may be attempted by that party by assuming the identity of another party that has the appropriate authorizations.

Two secondary threats are also identified. The security protocols defined in this memo do provide protection against:

Message Stream Modification

The SNMPv2 protocol is typically based upon a connectionless transport service which may operate over any subnetwork service. The re-ordering, delay or replay of messages can and does occur through the natural operation of many such subnetwork services. The message stream modification threat is the danger that messages may be maliciously re-ordered, delayed or replayed to an extent which is greater than can occur through the natural operation of a subnetwork service, in order to effect unauthorized management operations.

Disclosure

The disclosure threat is the danger of eavesdropping on the exchanges between managed agents and a management station. Protecting against this threat may be required as a matter of local policy.

There are at least two threats that an SNMPv2 security protocol need not protect against. The security protocols defined in this memo do not provide protection against:

Denial of Service

An SNMPv2 security protocol need not attempt to address the broad range of attacks by which service to authorized parties is denied. Indeed, such denial-of-service attacks are in many cases indistinguishable from the type of network failures with which any viable network management protocol must cope as a matter of course.

Traffic Analysis

In addition, an SNMPv2 security protocol need not attempt to address traffic analysis attacks. Indeed, many traffic patterns are predictable - agents may be managed on a regular basis by a relatively small number of management stations - and therefore there is no significant advantage afforded by protecting against traffic analysis.

1.3. Goals and Constraints

Based on the foregoing account of threats in the SNMP network management environment, the goals of an SNMPv2 security protocol are enumerated below.

(1) The protocol should provide for verification that each received SNMPv2 message has not been modified during its transmission

through the network in such a way that an unauthorized management operation might result.

(2) The protocol should provide for verification of the identity of the originator of each received SNMPv2 message.

(3) The protocol should provide that the apparent time of generation for each received SNMPv2 message is recent.

(4) The protocol should provide, when necessary, that the contents of each received SNMPv2 message are protected from disclosure.

In addition to the principal goal of supporting secure network management, the design of any SNMPv2 security protocol is also influenced by the following constraints:

(1) When the requirements of effective management in times of network stress are inconsistent with those of security, the design should prefer the former.

(2) Neither the security protocol nor its underlying security mechanisms should depend upon the ready availability of other network services (e.g., Network Time Protocol (NTP) or secret/key management protocols).

(3) A security mechanism should entail no changes to the basic SNMP network management philosophy.

1.4. Security Services

The security services necessary to support the goals of an SNMPv2 security protocol are as follows.

Data Integrity
: is the provision of the property that data has not been altered or destroyed in an unauthorized manner, nor have data sequences been altered to an extent greater than can occur non-maliciously.

Data Origin Authentication
: is the provision of the property that the claimed origin of received data is corroborated.

Data Confidentiality
: is the provision of the property that information is not made

available or disclosed to unauthorized individuals, entities, or processes.

The protocols specified in this memo require both data integrity and data origin authentication to be used at all times. For these protocols, it is not possible to obtain data integrity without data origin authentication, nor is it possible to obtain data origin authentication without data integrity.

Further, there is no provision for data confidentiality without both data integrity and data origin authentication.

1.5. Mechanisms

The security protocols defined in this memo employ several types of mechanisms in order to realize the goals and security services described above:

- In support of data integrity, a message digest algorithm is required. A digest is calculated over an appropriate portion of an SNMPv2 message and included as part of the message sent to the recipient.

- In support of data origin authentication and data integrity, the portion of an SNMPv2 message that is digested is first prefixed with a secret value shared by the originator of that message and its intended recipient.

- To protect against the threat of message delay or replay, (to an extent greater than can occur through normal operation), a timestamp value is included in each message generated. A recipient evaluates the timestamp to determine if the message is recent. This protection against the threat of message delay or replay does not imply nor provide any protection against unauthorized deletion or suppression of messages. Other mechanisms defined independently of the security protocol can also be used to detect message replay (e.g., the request-id [12]), or for set operations, the re-ordering, replay, deletion, or suppression of messages (e.g., the MIB variable snmpSetSerialNo [14]).

- In support of data confidentiality, a symmetric encryption algorithm is required. An appropriate portion of the message is encrypted prior to being transmitted to its recipient.

The security protocols in this memo are defined independently of the particular choice of a message digest and encryption algorithm - owing principally to the lack of a suitable metric by which to evaluate the security of particular algorithm choices. However, in the interests of completeness and in order to guarantee interoperability, Sections 1.5.1 and 1.5.2 specify particular choices, which are considered acceptably secure as of this writing. In the future, this memo may be updated by the publication of a memo specifying substitute or alternate choices of algorithms, i.e., a replacement for or addition to the sections below.

1.5.1. Message Digest Algorithm

In support of data integrity, the use of the MD5 [3] message digest algorithm is chosen. A 128-bit digest is calculated over the designated portion of an SNMPv2 message and included as part of the message sent to the recipient.

An appendix of [3] contains a C Programming Language implementation of the algorithm. This code was written with portability being the principal objective. Implementors may wish to optimize the implementation with respect to the characteristics of their hardware and software platforms.

The use of this algorithm in conjunction with the Digest Authentication Protocol (see Section 3) is identified by the ASN.1 object identifier value v2md5AuthProtocol, defined in [4].

For any SNMPv2 party for which the authentication protocol is v2md5AuthProtocol, the size of its private authentication key is 16 octets.

Within an authenticated management communication generated by such a party, the size of the authDigest component of that communication (see Section 3) is 16 octets.

1.5.2. Symmetric Encryption Algorithm

In support of data confidentiality, the use of the Data Encryption Standard (DES) in the Cipher Block Chaining mode of operation is chosen. The designated portion of an SNMPv2 message is encrypted and included as part of the message sent to the recipient.

Two organizations have published specifications defining the DES: the National Institute of Standards and Technology (NIST) [5] and the American National Standards Institute [6]. There is a companion Modes of Operation specification for each definition (see [7] and [8], respectively).

The NIST has published three additional documents that implementors may find useful.

- There is a document with guidelines for implementing and using the DES, including functional specifications for the DES and its modes of operation [9].

- There is a specification of a validation test suite for the DES [10]. The suite is designed to test all aspects of the DES and is useful for pinpointing specific problems.

- There is a specification of a maintenance test for the DES [11]. The test utilizes a minimal amount of data and processing to test all components of the DES. It provides a simple yes-or-no indication of correct operation and is useful to run as part of an initialization step, e.g., when a computer reboots.

The use of this algorithm in conjunction with the Symmetric Privacy Protocol (see Section 4) is identified by the ASN.1 object identifier value desPrivProtocol, defined in [4].

For any SNMPv2 party for which the privacy protocol is desPrivProtocol, the size of the private privacy key is 16 octets, of which the first 8 octets are a DES key and the second 8 octets are a DES Initialization Vector. The 64-bit DES key in the first 8 octets of the private key is a 56 bit quantity used directly by the algorithm plus 8 parity bits - arranged so that one parity bit is the least significant bit of each octet. The setting of the parity bits is ignored by the desPrivProtocol.

The length of the octet sequence to be encrypted by the DES must be an integral multiple of 8. When encrypting, the data should be padded at the end as necessary; the actual pad value is irrelevant.

If the length of the octet sequence to be decrypted is not an integral multiple of 8 octets, the processing of the octet sequence should be halted and an appropriate exception noted. Upon decrypting, the padding should be ignored.

2. SNMPv2 Party

Recall from [1] that:

- an SNMPv2 entity is an SNMPv2 protocol implementation used by an SNMPv2 agent and/or by one or more management applications;

- an SNMPv2 party is an identity assumed by an SNMPv2 entity in order to restrict its actions (for security or other purposes) to an administratively defined subset of all SNMPv2 possible actions;

- each SNMPv2 entity maintains a local database, called the Local Party Datastore (LPD), in which it retains its own set of information about local and remote SNMPv2 parties and other associated (e.g., access control) information.

- each SNMPv2 party has a set of attributes, the generic significance of which is defined in [1].

For each SNMPv2 party that supports the Digest Authentication Protocol, some attributes have additional, specific significance:

partyAuthProtocol -
the party's authentication protocol which identifies a combination of the Digest Authentication Protocol with a particular digest algorithm (such as that defined in Section 1.5.1). This combined mechanism is used to authenticate the origin and integrity of all messages generated by the party.

partyAuthClock -
the party's authentication clock which is the local notion of the current time that is specific to the party.

partyAuthPrivate -
the party's private authentication key which represents the secret value needed to support the Digest Authentication Protocol and associated digest algorithm.

partyAuthPublic -
the party's public authentication key which represents any public value that may be needed to support the authentication protocol. This attribute is not significant except as suggested in Section 5.4.

partyAuthLifetime -
the lifetime which represents an administrative upper bound on acceptable delivery delay for protocol messages generated by the party.

For each SNMPv2 party that supports the Symmetric Privacy Protocol, some attributes have additional, specific significance:

partyPrivProtocol -
the party's privacy protocol which identifies a combination of the Symmetric Privacy Protocol with a particular encryption algorithm (such as that defined in Section 1.5.2). This combined mechanism is used to protect from disclosure all protocol messages received by the party.

partyPrivPrivate -
the party's private privacy key which represents any secret value needed to support the Symmetric Privacy Protocol and associated encryption algorithm.

partyPrivPublic -
the party's public privacy key which represents any public value that may be needed to support the privacy protocol. This attribute is not significant except as suggested in Section 5.4.

3. Digest Authentication Protocol

The Digest Authentication Protocol provides both for verifying the integrity of a received message (i.e., the message received is the message sent) and for verifying the origin of a message (i.e., the reliable identification of the originator). The integrity of the message is protected by computing a digest over an appropriate portion of a message. The digest is computed by the originator of the message, transmitted with the message, and verified by the recipient of the message.

A secret value known only to the originator and recipient of the message is prefixed to the message prior to the digest computation. Thus, the origin of the message is known implicitly with the verification of the digest.

A requirement on parties using this Digest Authentication Protocol is that they shall not originate messages for transmission to any receiving party which does not also use this Digest Authentication Protocol. This restriction excludes undesirable side effects of communication between a party which uses these security protocols and a party which does not.

Recall from [1] that an SNMPv2 management communication is an ASN.1 value with the following syntax:

```
     SnmpMgmtCom ::= [2] IMPLICIT SEQUENCE {
       dstParty
          OBJECT IDENTIFIER,
       srcParty
          OBJECT IDENTIFIER,
       context
          OBJECT IDENTIFIER,
       pdu
          PDUs
     }
```

where:

dstParty -
   the destination SNMPv2 party of the SNMPv2 message.

srcParty -
   the source SNMPv2 party of the SNMPv2 message.

context -
the SNMPv2 context containing the management information referenced by the SNMPv2 message.

pdu -
an SNMPv2 PDU as defined in [12].

Also recall from [1] that an SNMPv2 authenticated management communication is an ASN.1 value with the following syntax:

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

where:

authInfo -
the authentication information required in support of the authentication protocol used by the source SNMPv2 party. The detailed significance of the authentication information is specific to the authentication protocol in use, and its only use is in determining whether the communication is authentic or not.

authData -
an SNMPv2 management communication.

In support of the Digest Authentication Protocol, an authInfo component is of type AuthInformation:

```
AuthInformation ::= [2] IMPLICIT SEQUENCE {
  authDigest
     OCTET STRING,
  authDstTimestamp
     UInteger32 (0..2147483647),
  authSrcTimestamp
     UInteger32 (0..2147483647)
}
```

where:

authDigest -
the authentication digest which is the value of the digest computed over an appropriate portion of the message, where the message is temporarily prefixed with a secret value for the purposes of computing the digest.

authSrcTimestamp -
the source authentication timestamp which represents the time of the generation of the message according to the partyAuthClock of the source SNMPv2 party. Note that the granularity of the authentication timestamp is 1 second.

authDstTimestamp -
the destination authentication timestamp which represents the time of the generation of the message according to the partyAuthClock of the destination SNMPv2 party. Note that the granularity of the authentication timestamp is 1 second.

3.1. Generating a Message

This section describes the behavior of an SNMPv2 entity when it assumes the identity of an SNMPv2 party using the Digest Authentication Protocol in order to generate a message. Since the behavior of an SNMPv2 entity when generating protocol messages is defined generically in [1], only those aspects of that behavior that are specific to the Digest Authentication Protocol are described below. In particular, this section describes the encapsulation of an SNMPv2 management communication into an SNMPv2 authenticated management communication.

According to Section 3.1 of [1], a SnmpAuthMsg value is constructed during Step 3 of generic processing. When the authentication protocol is the Digest Authentication Protocol, the procedure is as follows.

(1) The LPD is consulted to retrieve the current values of the authentication clock and private authentication key of the source SNMPv2 party. The LPD is also consulted to retrieve the current value of the authentication clock of the destination SNMPv2 party.

(2) The authSrcTimestamp component is set to the retrieved authentication clock value of the source SNMPv2 party. The authDstTimestamp component is set to the retrieved authentication clock value of the destination SNMPv2 party.

(3) The authDigest component is temporarily set to the private authentication key of the source SNMPv2 party. The SnmpAuthMsg value is serialized (i.e., encoded) according to the conventions of [13] and [12]. A digest is computed over the octet sequence representing the serialized SnmpAuthMsg value using, for example, the algorithm specified in Section 1.5.1. The authDigest component is then set to the computed digest value.

As set forth in [1], the SnmpAuthMsg value is then encapsulated according to the appropriate privacy protocol into a SnmpPrivMsg value. This latter value is then serialized and transmitted to the destination SNMPv2 party.

3.2. Receiving a Message

This section describes the behavior of an SNMPv2 entity upon receipt of a protocol message from an SNMPv2 party using the Digest Authentication Protocol. Since the behavior of an SNMPv2 entity when receiving protocol messages is defined generically in [1], only those aspects of that behavior that are specific to the Digest Authentication Protocol are described below.

According to Section 3.2 of [1], a SnmpAuthMsg value is evaluated during Step 9 of generic processing. When the authentication protocol is the Digest Authentication Protocol, the procedure is as follows.

(1) If the ASN.1 type of the authInfo component is not AuthInformation, then the message is evaluated as unauthentic, the snmpStatsBadAuths counter [14] is incremented, and a report PDU [1] is generated. Otherwise, the authSrcTimestamp, authDstTimestamp, and authDigest components are extracted from the SnmpAuthMsg value.

(2) The LPD is consulted to retrieve the current values of the authentication clock, private authentication key and lifetime of the source SNMPv2 party.

(3) If the authSrcTimestamp component plus the lifetime is less than the retrieved authentication clock, then the message is evaluated as unauthentic, the snmpStatsNotInLifetimes counter [14] is incremented, and a report PDU [1] is generated.

(4) The authDigest component is extracted and temporarily recorded.

(5) A new SnmpAuthMsg value is constructed such that its authDigest component is set to the private authentication key and its other components are set to the value of the corresponding components in the received SnmpAuthMsg value. This new SnmpAuthMsg value is serialized according to the conventions of [13] and [12]. A digest is computed over the octet sequence representing that serialized value using, for example, the algorithm specified in Section 1.5.1.

NOTE

Because serialization rules are unambiguous but may not be unique, great care must be taken in reconstructing the serialized value prior to computing the digest. Implementations may find it useful to keep a copy of the original serialized value and then simply modify the octets which directly correspond to the placement of the authDigest component, rather than re-applying the serialization algorithm to the new SnmpAuthMsg value.

(6) If the computed digest value is not equal to the digest value temporarily recorded in step 4 above, the message is evaluated as unauthentic, then the snmpStatsWrongDigestValues counter [14] is incremented and a report PDU [1] is generated.

Otherwise, the message is evaluated as authentic.

(7) The LPD is consulted for access privileges permitted by local access policies for the given source destination SNMPv2 parties. If any level of access is permitted, then the Selective Clock Acceleration mechanism is invoked as follows:

if the authSrcTimestamp value is greater than the current value of the authentication clock stored in the LPD for the source SNMPv2 party, then that current value is advanced to the authSrcTimestamp value; and,

if the authDstTimestamp value is greater than the current value of the authentication clock stored in the LPD for the destination SNMPv2 party, then that current value is advanced to the authDstTimestamp value.

(Note that this step is conceptually independent from Steps 15-17 of Section 3.2 in [1]).

If the SnmpAuthMsg value is evaluated as unauthentic, an authentication failure is noted and the received message is discarded without further processing. Otherwise, processing of the received message continues as specified in [1].

4. Symmetric Privacy Protocol

The Symmetric Privacy Protocol provides for an appropriate portion of a message to be encrypted according to a secret key known only to the originator and recipient of the message.

This protocol assumes the underlying mechanism is a symmetric encryption algorithm. In addition, the message to be encrypted must be protected according to the conventions of the Digest Authentication Protocol.

Recall from [1] that an SNMPv2 private management communication is an ASN.1 value with the following syntax:

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

where:

privDst -
the destination SNMPv2 party of the SNMPv2 message.

privData -
the (possibly encrypted) serialization (i.e., encoding according to the conventions of [13] and [12]) of an SNMPv2 authenticated management communication.

4.1. Generating a Message

This section describes the behavior of an SNMPv2 entity when it generates a message having a destination SNMPv2 party which uses the Symmetric Privacy Protocol. Since the behavior of an SNMPv2 entity when generating a protocol message is defined generically in [1], only those aspects of that behavior that are specific to the Symmetric Privacy Protocol are described below. In particular, this section describes the encapsulation of an SNMPv2 authenticated management communication into an SNMPv2 private management communication.

According to Section 3.1 of [1], a SnmpPrivMsg value is constructed during Step 5 of generic processing. When the privacy protocol is the Symmetric Privacy Protocol, the procedure is as follows.

(1) If the SnmpAuthMsg value is not authenticated according to the conventions of the Digest Authentication Protocol, the generation of the private management communication fails without further processing.

(2) The LPD is consulted to retrieve the private privacy key of the destination SNMPv2 party.

(3) The SnmpAuthMsg value is serialized according to the conventions of [13] and [12].

(4) The octet sequence representing the serialized SnmpAuthMsg value is encrypted using, for example, the algorithm specified in Section 1.5.2 and the retrieved private privacy key.

(5) The privData component is set to the encrypted value.

As set forth in [1], the SnmpPrivMsg value is then serialized and transmitted to the destination SNMPv2 party.

4.2. Receiving a Message

This section describes the behavior of an SNMPv2 entity upon receipt of a protocol message having a destination SNMPv2 party which uses the Symmetric Privacy Protocol. Since the behavior of an SNMPv2 entity when receiving a protocol message is defined generically in [1], only those aspects of that behavior that are specific to the Symmetric Privacy Protocol are described below.

According to Section 3.2 of [1], the privData component of a received SnmpPrivMsg value is evaluated during Step 4 of generic processing. When the privacy protocol is the Symmetric Privacy Protocol, the procedure is as follows.

(1) The LPD is consulted to retrieve the private privacy key of the destination SNMPv2 party.

(2) The contents octets of the privData component are decrypted using, for example, the algorithm specified in Section 1.5.2 and the retrieved private privacy key.

Processing of the received message continues as specified in [1].

5. Clock and Secret Distribution

The protocols described in Sections 3 and 4 assume the existence of loosely synchronized clocks and shared secret values. Three requirements constrain the strategy by which clock values and secrets are distributed.

- If the value of an authentication clock is decreased, the private authentication key must be changed concurrently.

  When the value of an authentication clock is decreased, messages that have been sent with a timestamp value between the value of the authentication clock and its new value may be replayed and appear to be recently generated. Changing the private authentication key ensures that such replays will not be evaluated as authentic.

- The private authentication key and private privacy key must be known only to the parties requiring knowledge of them.

  Protecting the secrets from disclosure is critical to the security of the protocols. Knowledge of the secrets must be as restricted as possible within an implementation. A management station has the additional responsibility of remembering the state of all parties across reboots, e.g., by recording the secrets on a long-term storage device. Access to information on this device must be as restricted as is practically possible.

- There must exist at least one SNMPv2 entity that assumes the role of a responsible management station.

  This management station is responsible for ensuring that all authentication clocks are synchronized and for changing the secret values when necessary. Although more than one management station may share this responsibility, their coordination is essential to the secure management of the network. The mechanism by which multiple management stations ensure that no more than one of them attempts to synchronize the clocks or update the secrets at any one time is a local implementation issue.

  A responsible management station may either support clock synchronization and secret distribution as separate functions, or combine them into a single functional unit.

5.1. Initial Configuration

This section describes the initial configuration of an SNMPv2 entity that supports the Digest Authentication Protocol or both the Digest Authentication Protocol and the Symmetric Privacy Protocol.

When a network device is first installed, its initial, secure configuration must be done manually, i.e., a person must physically visit the device (or perhaps, the device is initially configured on an secure, physically-isolated network), and enter the initial secret values for at least its first pair of secure SNMPv2 parties.

Requiring a person to physically visit a device every time an SNMPv2 party is configured is administratively prohibitive. The recommended procedure is to configure a small set of initial SNMPv2 parties for each SNMPv2 entity, one pair of which may be subsequently used to configure all other SNMPv2 parties.

Configuring one pair of SNMPv2 parties which are then used to configure all other parties has the advantage of exposing (to humans) only one pair of secrets. To limit this exposure, the responsible management station should change these values as its first operation upon completion of the initial configuration. In this way, secrets are known only to the peers requiring knowledge of them in order to communicate. For enhanced security, it is further recommended that this one pair of parties not be used for any purpose other than the configuration of other SNMPv2 parties.

The minimal, useful set of SNMPv2 parties that can be configured for each managed agent is four parties, one of each of the following for both the responsible management station and the managed agent:

- an SNMPv2 party for which the authentication protocol and privacy protocol are noAuth and noPriv [1], respectively, and

- an SNMPv2 party for which the authentication protocol identifies the mechanism defined in Section 1.5.1, and its privacy protocol is either: noPriv, or the mechanism defined in Section 1.5.2. (Note that the use of a privacy protocol other than noPriv provides a greater level of security, but is not required by this specification.)

When installing a managed agent, these parties need to be configured with their initial secrets, etc., both in the responsible management station and in the new agent. If the responsible management station is

configured first, it can be used to generate the initial secrets and provide them to a person for distribution to the managed agent. The following sequence of steps describes the initial configuration of a managed agent and its responsible management station.

(1) Determine the initial values for each of the attributes of the SNMPv2 party to be configured. Some of these values may be specified in the MIB document, some may be administratively determined, and some may be computed by the responsible management station.

(2) Configure the parties in the responsible management station, recording any initial values computed by the management station.

(3) Configure the parties in the managed agent, according to the set of initial values.

(4) The responsible management station must then synchronize the authentication clock values for each party it shares with each managed agent. Section 5.3 specifies one strategy by which this could be accomplished.

(5) The responsible management station should change the secret values manually configured to ensure the actual values are known only to the peers requiring knowledge of them in order to communicate. To do this, the management station generates new secrets for each party to be reconfigured and distributes the updates using any strategy which protects the new values from disclosure (e.g., the recommended strategy and procedure described in section 5.4). Upon receiving positive acknowledgement that the new values have been distributed, the management station should update its LPD with the new values.

If there are other SNMPv2 entities requiring knowledge of the secrets, the responsible management station must distribute the information upon completion of the initial configuration. The considerations, mentioned above, concerning the protection of secrets from disclosure, also apply in this case.

5.2. Clock Distribution

For each SNMPv2 party using the Digest Authentication Protocol, the responsible management station must ensure that the party's authentication clock value

- is loosely synchronized among all the LPDs in which it appears, |
- is reset, as indicated below, upon reaching its maximal value, and
- is non-decreasing, except as indicated below.

The skew among the clock values must be accounted for in the lifetime value, in addition to the expected communication delivery delay.

A skewed authentication clock may be detected by a number of strategies, including knowledge of the accuracy of the system clock, unauthenticated queries of the appropriate MIB object [4], and recognition of authentication failures originated by the party. A procedure for correcting clock skew is presented in Section 5.3.

Every SNMPv2 entity must react correctly as an authentication clock approaches its maximal value. If the authentication clock for a particular SNMPv2 party ever reaches the maximal time value, the clock must halt at that value. (The value of interest may be the maximum less lifetime. When authenticating a message, its authentication timestamp is added to lifetime and compared to the authentication clock. An SNMPv2 entity must guarantee that the sum is never greater than the maximal time value.) In this state, the only authenticated request a management station should generate for this party is one that alters the value of at least its authentication clock and private authentication key. In order to reset these values, the responsible management station may set the authentication timestamp in the message to the maximal time value.

The value of the authentication clock for a particular SNMPv2 party must never be altered such that its new value is less than its old value, unless its private authentication key is also altered at the same time.

5.3. Clock Synchronization

Unless the secrets are changed at the same time, the correct way to synchronize clocks is to advance the slower clock to be equal to the faster clock.

Suppose that agentParty is a party in a managed agent, and that mgrParty is a party in a corresponding responsible management station. Then, there are four possible conditions of the authentication clocks that could require correction:

(1) The management station's notion of the value of agentParty's authentication clock is greater than the agent's notion.

(2) The management station's notion of the value of mgrParty's authentication clock is greater than the agent's notion.

(3) The agent's notion of the value of agentParty's authentication clock is greater than the management station's notion.

(4) The agent's notion of the value of mgrParty's authentication clock is greater than the management station's notion.

The Selective Clock Acceleration mechanism described in step 7 of Section 3.2 corrects conditions 1, 2 and 3 as part of the normal processing of an authentic message. Thus, the clock adjustment procedure below need not provide for any adjustments in those cases. Rather, the following sequence of steps specifies how the clocks may be synchronized when condition 4 occurs.

(1) The responsible management station retrieves the authentication clock value for mgrParty from the agent. This retrieval must be an unauthenticated request, since the management station knows/suspects that the clocks are unsynchronized. If the request fails, the clocks cannot be synchronized, and the clock adjustment procedure is aborted without further processing.

(2) If the notion of the authentication clock for mgrParty just retrieved from the agent exceeds the management station's notion, then condition 4 is manifest, and the responsible management station advances its notion of the authentication clock for mgrParty to match the agent's notion.

By virtue of the mandatory support for maintenance functions using internalParty and internalContext [1], the retrieval performed in Step 1 above SHOULD be performed with these parameters:

```
srcParty   = internalParty
dstParty   = internalParty
context    = internalContext
operation  = get, get-next, or get-bulk
```

A potential operational problem is the rejection of authentic management operations which were authenticated using a previous value of the relevant party clock. This possibility may be avoided if a management station defers generation of management traffic between relevant parties while this clock adjustment procedure is in progress.

When authenticated parties are used for sending traps, the above steps will not be performed by the agent, nor will the agent detect unsynchronized clock conditions. Rather, the manager must perform these functions, either as a by-product of using the same set of parties for other management operations, or by periodic use of these parties just for this purpose. (Note that periodic communication with the agent using these parties can be used not only as a means to keep the clocks synchronized, but also as a means to detect agent/network failures which also affect the delivery of such traps.)

Advancement of the value of a clock as described above does not introduce any new security vulnerabilities since the value of the clock is intended to increase with the passage of time. While it is possible for an attacker to respond to the unauthenticated retrieval with a clock value more advanced than that held by the agent, the first succeeding exchange of messages between the manager and the agent negates any benefit in doing so, with the possible exception that it causes the party's clock to reach its maximum value sooner, a form of denial of service against which protection is not provided.

In general, the advancement of clock values by the manager except as required by the synchronization algorithm above (or its equivalent) or as required by the Selective Clock Acceleration mechanism (see step 7 of Section 3.2), is termed artificial clock advancement of the clock and strongly discouraged.

5.4. Secret Distribution

This section describes one strategy by which SNMPv2 entities which support at least the Digest Authentication Protocol can change the secrets for a particular SNMPv2 party. This strategy makes use of the MIB [4] which provides access via SNMPv2 messages to the parameters of all SNMPv2 parties known to an SNMPv2 entity.

The frequency with which the secrets of an SNMPv2 party should be changed is a local administrative issue. However, the more frequently a secret is used, the more frequently it should be changed. At a minimum, the secrets must be changed whenever the associated authentication clock approaches its maximal value. Note that, owing to automatic advances of the authentication clock described in this memo, the authentication clock for an SNMPv2 party may well approach its maximal value sooner than might otherwise be expected.

One of the capabilities provided by [4] is the ability to change the value of a party's private authentication key and/or private privacy key, through the use of an SNMPv2 set operation upon the relevant MIB object(s). This SNMPv2 set operation MUST be encapsulated in an SNMPv2 message having source and destination parties which use the Digest Authentication Protocol.

In addition, a mechanism is needed to prevent the disclosure of the (new or old) values to eavesdroppers. To achieve this, [4] defines two mechanisms.

First, the values of both the authentication key and the privacy key are not directly represented as MIB objects. Rather, they are represented according to an encoding. This encoding is the bitwise exclusive-OR of the old key value with the new key value. Use of this encoding provides that neither the new value nor the old value are exposed even if a SNMPv2 set operation to modify the values directly is encapsulated in an SNMPv2 message having source and destination parties which use the noPriv protocol. In addition, it allows the value of the party's authentication key to be changed, thereby allowing the value of the party's authentication clock to be decreased (e.g., when approaching its maximal value). However, it does nothing to prevent an eavesdropper with knowledge of the old value from immediately computing the new value.

Second, [4] defines a mechanism whereby the concatenation of an unpredictable value and the old key is subjected to the MD5 one-way hash algorithm and the resulting digest exclusive-OR-ed with a pre-computed 'delta' value to obtain a desired value for the new key (see the recommended procedure below). Since this second mechanism employs a one-way function, an attacker who learns the new value of the secret cannot calculate the previous value, even if he has observed all the management traffic used to make the change.

Nevertheless, there are times (e.g., when there is a possibility that the old value of a secret has been compromised) when a network administration may require the assured ability to set a new uncompromised value. Such a requirement can normally be met by using source and destination parties which use encryption (e.g., the Symmetric Privacy Protocol) for the SNMPv2 set operation. Of course, if the privacy key of either of those parties is also compromised, the eavesdropper can still easily compute the new value. Alternatively, a party whose secrets are possibly compromised can be destroyed, and recreated (see section 5.5) based on the secrets of some other uncompromised party.

When an SNMPv2 party is used to change its own secrets (e.g., using the maintenance function and recommended procedure detailed below) or to destroy itself, the destination is required to change the secret value(s) stored in its LPD after generating its response. Thus, the response will be constructed according to the old value. As the responsible management station will not update its LPD until after the response is received, it is able to interpret unambiguously any response it receives regardless of whether an error occurred in processing the set operation.

By virtue of the mandatory support for maintenance functions on the internalContext [1], the set operation to change the secrets of an existing party, either <mgrParty>, a party in the manager, or <agentParty>, a party in an agent, MAY be generated with these parameters:

```
srcParty   = <mgrParty>
dstParty   = <agentParty>
context    = internalContext
operation  = set
```

to update the secrets within the conceptual row in the partyTable which corresponds to either of the authenticated parties, <mgrParty> or <agentParty>.

This operation is authorized for any combination of <mgrParty> and <agentParty> for which at least one ACL exists having:

acTarget	<agentParty>	+
acSubject	<mgrParty>	+
acContext	any	+

When such an authorized combination of <mgrParty> and <agentParty> is used to access <internalContext>, the operation uses a pseudo-ACL:

acTarget	<agentParty>	+
acSubject	<mgrParty>	+
acContext	internalContext	+
acReadViewIndex	<pseudo-view>	+
acWriteViewIndex	<pseudo-view>	+
acPrivileges	43 (Get, GetNext, GetBulk, Set)	+

where <pseudo-view> is statically defined to be exactly the following subtrees:

partyAuthClock.agentParty	+
partyAuthPrivate.agentParty	+
partyAuthPublic.agentParty	+
partyPrivPrivate.agentParty	+
partyPrivPublic.agentParty	+
partyAuthChange.agentParty	+
partyPrivChange.agentParty	+
partyAuthClock.mgrParty	+
partyAuthPrivate.mgrParty	+
partyAuthPublic.mgrParty	+
partyPrivPrivate.mgrParty	+
partyPrivPublic.mgrParty	+
partyAuthChange.mgrParty	+
partyPrivChange.mgrParty	+
partySecretSpinLock	+

After transmitting an SNMPv2 set operation to request the value of a secret be changed, if the responsible management station does not receive a response, there are two possible causes.

- The request may not have been delivered to the destination.

- The response may not have been delivered to the originator of the request.

In order to distinguish these two possible error conditions, a responsible management station could check the destination to see if the change has occurred. Unfortunately, since the secret values are unreadable, this is not directly possible.

The recommended strategy for verifying key changes is to set the public value corresponding to the secret being changed to a recognizable, novel value: that is, alter the public authentication key value for the relevant party when changing its private authentication key, or alter its public privacy key value when changing its private privacy key. In this way, the responsible management station may retrieve the public value when a response is not received, and verify whether or not the change has taken place. (This strategy is available since the public values are not used by the protocols defined in this memo. If this strategy is employed, then the public values do become significant. Of course, protocols using the public values can make use of this strategy directly.)

In particular, the following procedure is recommended to change the secrets of a pair of parties (note that if a party's secrets are compromised, some other party should be used to perform these operations):

(1) The management station determines the values for the new secrets, generates an unpredictable value for each, and computes the appropriate delta values:

```
determine desired values for key1New and key2New
randomValue1 = unpredictable()
randomValue2 = unpredictable()
keyIntermediate1 = md5(key1Old || randomValue1)
deltaValue1 = key1New XOR keyIntermediate1
keyIntermediate2 = md5(key2Old || randomValue2)
deltaValue2 = key2New XOR keyIntermediate2
```

(2) The management station initialises its knowledge of the current state of the agent using an authenticated get operation, retrying as necessary until a response is received:

```
get (
    lastLock = partySecretSpinLock.0,
    lastNovel = partyAuthPublic.<party1>
)
```

(3) The management station generates a unique novel value (which must be different from all previous values of lastNovel used with these new secret values). It then concatenates the unpredictable and delta values for each party, and conveys them to the agent in a single varbindlist, together with the most recently retrieved value of the advisory lock and the most recently generated unique novel value, using an authenticated set operation with a previously unused value of request-id.

```
set (
    partySecretSpinLock.0 = lastLock,
    partyAuthChange.<party1> = <randomValue1 || deltaValue1>,
    partyAuthChange.<party2> = <randomValue2 || deltaValue2>,
    partyAuthPublic.<party1> = uniqueNovelValue
)
```

If a successful response with the correct request-id value is received, then goto step 4.

If no response or an error response (with the correct request-id) is received, then the operation may or may not have been successful, due to duplication and/or loss of the request and/or the response(s). So,

- save the error-index and error-status values,
- re-issue the get operation in step 2; if <party1> and <party2> are being used to change their own secrets, then this operation is performed using the new secrets, key1New and key2New, and if this operation fails due to an increment of snmpStatsWrongDigestValues [4], then goto step 2;
- retry this get operation as necessary until a response is received,
- if this response indicates that partyAuthPublic has the unique novel value assigned in the last set operation, goto step 4.

Otherwise, the set operation failed, and the saved error values are inspected to determine the cause of the failure.

- if no response was received or the error-index indicates a problem with partySecretSpinLock, goto step 2.
- if the error-index indicates a problem with partyAuthChange or partyAuthPublic, the secret cannot be changed to the new value.

(4) Record the new secret values in stable storage. The operation is now successfully completed.

[Retry counts to prevent endlessly looping in the presence of certain failures were omitted from the above procedure in the interest of brevity.]

Note that during the period of time after the request has been sent and before the success of the operation is determined, the management station must keep track of both the old and new secret values. Since the delay may be the result of a network failure, the management station must be prepared to retain both values for an extended period of time, including across reboots.

5.5. Party Cloning

Another capability provided by [4] is the ability to create a new SNMPv2 party as a "clone" of an existing party. That is, when a new party is created, if values are not specified for any of its authentication and privacy parameters, then those parameters take the same values as those of the party being "cloned". Specifically, the parameters are the authentication protocol, the public authentication key, the lifetime, the privacy protocol and the public privacy key. In addition, the appropriate values of the party being cloned are used as the initial "old key" values of the new party's private authentication key and the private privacy key for the purposes of the exclusive-OR encoding associated with an SNMPv2 set operation.

Further, the new party is not allowed to become active until SNMPv2 set operation(s) have not only specified the party being cloned but also changed the new party's private authentication key and private privacy key.

Similar considerations to those mentioned in the preceding section on exposure of key values and on the possibility of compromised values,

also apply in this situation.

5.6. Crash Recovery

This section describes the requirements for SNMPv2 entities in connection with recovery from system crashes or other service interruptions.

For each SNMPv2 party in the LPD for a particular SNMPv2 entity, the SNMPv2 entity must retain the values of the following information in non-volatile storage:

- its identity,
- an indication of its authentication and privacy protocols,
- its authentication clock,
- its lifetime,
- its private authentication key, and
- its private privacy key.

The authentication clock of an SNMPv2 party is a critical component of the overall security of the protocols. The inclusion of a reliable representation of a clock in an SNMPv2 entity is required for overall security. A reliable clock representation ensures that a clock's value is monotonically increasing, even across a power loss or other system failure of the local SNMPv2 entity. One example of a reliable clock representation is that provided by battery-powered clock-calendar devices incorporated into some contemporary systems. Another example is storing and updating a clock value in non-volatile storage at a frequency of once per U (e.g., 24) hours, and re-initialising that clock value on every reboot as the stored value plus U hours. It is assumed that management stations always support reliable clock representations, where clock adjustment by a human operator during crash recovery may contribute to that reliability.

If a managed agent crashes and does not reboot in time for its responsible management station to prevent its authentication clock from reaching its maximal value, upon reboot the clock must be halted at its maximal value. The procedures specified in Section 5.3 would then apply.

6. Security Considerations

6.1. Recommended Practices

This section describes practices that contribute to the secure, effective operation of the mechanisms defined in this memo.

- A management station should discard SNMPv2 responses for which neither the request-id component nor the represented management information corresponds to any currently outstanding request.

  Although it would be typical for a management station to do this as a matter of course, when using these security protocols it is significant due to the possibility of message duplication (malicious or otherwise).

- A management station should not interpret an agent's lack of response to an authenticated SNMPv2 management communication as a conclusive indication of agent or network failure. Authentication clock skew or inconsistent notions of shared secrets can also result in a lack of response. To distinguish between these causes, an unauthenticated retrieval of the party clocks should be used; a lack of response to this unauthenticated request will indicate agent/network failure; alternatively, the retrieved clock values will reveal if clock skew has occurred. If neither of these conditions apply, then inspection of the agent's snmpStats counters [14] will reveal the cause.

- The lifetime value for an SNMPv2 party should be chosen by the local administration as a compromise between: a) the need to limit the size of the "window" during which replay is possible, and b) the accuracy of available clock mechanisms, the relevant round-trip communications delays, and the frequency with which a responsible management station will be able to verify all clock values. In particular, lifetime needs to be greater than all reasonable values of round-trip communications delay, including those at times of network stress conditions.

- When sending state altering messages to a managed agent, a management station should delay sending successive messages to the managed agent until a positive acknowledgement is received for the previous message or until the previous message expires.

  No message ordering is imposed by the SNMPv2. Messages may be

received in any order relative to their time of generation and each will be processed in the ordered received. Note that when an authenticated message is sent to a managed agent, it will be valid for a period of time that does not exceed lifetime under normal circumstances, and is subject to replay during this period.

Indeed, a management station must cope with the loss and re-ordering of messages resulting from anomalies in the network as a matter of course.

However, a managed object, snmpSetSerialNo [14], is specifically defined for use with SNMPv2 set operations in order to provide a mechanism to ensure the processing of SNMPv2 messages occurs in a specific order.

- The frequency with which the secrets of an SNMPv2 party should be changed is indirectly related to the frequency of their use.

  Protecting the secrets from disclosure is critical to the overall security of the protocols. Frequent use of a secret provides a continued source of data that may be useful to a cryptanalyst in exploiting known or perceived weaknesses in an algorithm. Frequent changes to the secret avoid this vulnerability.

  Changing a secret after each use is generally regarded as the most secure practice, but a significant amount of overhead may be associated with that approach.

  Note, too, in a local environment the threat of disclosure may be insignificant, and as such the changing of secrets may be less frequent. However, when public data networks are the communication paths, more caution is prudent.

- In order to foster the greatest degree of security, a management station implementation must support constrained, pairwise sharing of secrets among SNMPv2 entities as its default mode of operation.

  Owing to the use of symmetric cryptography in the protocols defined here, the secrets associated with a particular SNMPv2 party must be known to all other SNMPv2 parties with which that party may wish to communicate. As the number of locations at which secrets are known and used increases, the likelihood of their disclosure also increases, as does the potential impact of that disclosure. Moreover, if more than two SNMPv2 entities know a particular secret, then data origin cannot be reliably authenticated because

each such entity has the knowledge needed to generate a message which will be evaluated as authentic. Thus, the greatest degree of security is obtained through configurations in which the secrets for each SNMPv2 party are known to at most two SNMPv2 entities.

6.2. Conformance

An SNMPv2 entity implementation that claims conformance to this memo must satisfy the following requirements:

(1) It must implement the Digest Authentication Protocol in conjunction with the algorithm defined in Section 1.5.1.

(2) It must support maintenance operations using internalParty and internalContext [1]. |

(3) For each SNMPv2 party about which it maintains information in a LPD, an implementation must satisfy the following requirements: |

(a) It must not allow a party's parameters to be set to a value inconsistent with its expected syntax.

(b) It must, to the maximal extent possible, prohibit read-access to the private authentication key and private encryption key under all circumstances except as required to generate and/or validate SNMPv2 messages with respect to that party.

(c) It must allow the party's authentication clock to be publicly accessible. The correct operation of the Digest Authentication Protocol requires that it be possible to determine this value at all times in order to guarantee that skewed authentication clocks can be resynchronized.

(d) It must prohibit alterations to its local value of the party's authentication clock independently of alterations to its value of the private authentication key (unless the clock alteration is an advancement).

(e) It must never allow its local value of the party's authentication clock to be incremented beyond the maximal time value and so "roll-over" to zero.

(f) It must never increase its value of the party's lifetime except as may be explicitly authorized (via imperative command or securely represented configuration information) by the responsible network administrator.

(g) In the event that the non-volatile storage of a party's parameters (in particular, either the private authentication key or private encryption key) are lost or destroyed, it must alter the recorded values of these parameters to random values so subsequent interaction with that party requires manual redistribution of new secrets and other parameters.

(4) If it selects new value(s) for a party's secret(s), it must avoid bad or obvious choices for said secret(s). Choices to be avoided are boundary values (such as all-zeros) and predictable values (such as the same value as previously or selecting from a predetermined set).

(5) It must ensure that a received message for which the source party uses the Digest Authentication Protocol but the destination party does not, is always declared to be unauthentic. This may be achieved explicitly via an additional step in the procedure for processing a received message, or implicitly by verifying that all local access control policies enforce this requirement.

7. Acknowledgements

The authors wish to acknowledge the contributions of the SNMPv2 Working Group in general. In particular, the following individuals

Dave Arneson (Cabletron),
Uri Blumenthal (IBM),
Doug Book (Chipcom),
Maria Greene (Ascom Timeplex),
Deirdre Kostik (Bellcore),
Dave Harrington (Cabletron),
Jeff Johnson (Cisco Systems),
Brian O'Keefe (Hewlett Packard),
Dave Perkins (Bay Networks),
Randy Presuhn (Peer Networks),
Shawn Routhier (Epilogue),
Bob Stewart (Cisco Systems),
Kaj Tesink (Bellcore).

deserve special thanks for their contributions.

8. References

[1] Case, J., Galvin, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Administrative Infrastructure for Version 2 of the Simple Network Management Protocol (SNMPv2)", Internet Draft, SNMP Research, Inc., Trusted Information Systems, Cisco Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, March 1995.

[2] Case, J., Fedor, M., Schoffstall, M., Davin, J., "Simple Network Management Protocol", STD 15, RFC 1157, SNMP Research, Performance Systems International, MIT Laboratory for Computer Science, May 1990.

[3] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, MIT Laboratory for Computer Science, April 1992.

[4] Case, J., Galvin, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Party MIB for Version 2 of the Simple Network Management Protocol (SNMPv2)", Internet Draft, SNMP Research, Inc., Trusted Information Systems, Cisco Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, March 1995.

[5] Data Encryption Standard, National Institute of Standards and Technology. Federal Information Processing Standard (FIPS) Publication 46-1. Supersedes FIPS Publication 46, (January, 1977; reaffirmed January, 1988).

[6] Data Encryption Algorithm, American National Standards Institute. ANSI X3.92-1981, (December, 1980).

[7] DES Modes of Operation, National Institute of Standards and Technology. Federal Information Processing Standard (FIPS) Publication 81, (December, 1980).

[8] Data Encryption Algorithm - Modes of Operation, American National Standards Institute. ANSI X3.106-1983, (May 1983).

[9] Guidelines for Implementing and Using the NBS Data Encryption Standard, National Institute of Standards and Technology. Federal Information Processing Standard (FIPS) Publication 74, (April, 1981).

[10] Validating the Correctness of Hardware Implementations of the NBS Data Encryption Standard, National Institute of Standards and Technology. Special Publication 500-20.

[11] Maintenance Testing for the Data Encryption Standard, National Institute of Standards and Technology. Special Publication 500-61, (August, 1980).

[12] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", Internet Draft, SNMP Research, Inc., Cisco Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, March 1995.

[13] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", Internet Draft, SNMP Research, Inc., Cisco Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, March 1995.

[14] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)", Internet Draft, SNMP Research, Inc., Cisco Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, March 1995.

APPENDIX A - Protocol Correctness

The correctness of these SNMPv2 security protocols with respect to the goals stated in Section 1.4 depends on the following assumptions:

(1) The chosen message digest algorithm satisfies its design criteria. In particular, it is computationally infeasible to discover two messages that share the same digest value.

(2) It is computationally infeasible to determine the secret used in calculating a digest on the concatenation of the secret and a message when both the digest and the message are known.

(3) The chosen symmetric encryption algorithm satisfies its design criteria. In particular, it is computationally infeasible to determine the cleartext message from the ciphertext message without knowledge of the key used in the transformation.

(4) Local notions of a party's authentication clock while it is associated with a specific private key value are monotonically non-decreasing (i.e., they never run backwards) in the absence of administrative manipulations.

(5) The secrets for a particular SNMPv2 party are known only to authorized SNMPv2 entities.

(6) Local notions of the authentication clock for a particular SNMPv2 party are never altered such that the authentication clock's new value is less than the current value without also altering the private authentication key.

For each mechanism of the protocol, an informal account of its contribution to the required goals is presented below. Pseudocode fragments are provided where appropriate as examples of possible implementations; they are intended to be self-explanatory.

A.1 Clock Monotonicity Mechanism

By pairing each sequence of a clock's values with a unique key, the protocols partially realize goal 3, and the conjunction of this property with assumption 6 above is sufficient for the claim that, with respect to a specific private key value, all local notions of a party's authentication clock are, in general, non-decreasing with time.

A.2 Data Integrity Mechanism

The protocols require computation of a message digest computed over the SNMPv2 message prepended by the secret for the relevant party. By virtue of this mechanism and assumptions 1 and 2, the protocols realize goal 1.

Normally, the inclusion of the message digest value with the digested message would not be sufficient to guarantee data integrity, since the digest value can be modified in addition to the message while it is enroute. However, since not all of the digested message is included in the transmission to the destination, it is not possible to substitute both a message and a digest value while enroute to a destination.

Strictly speaking, the specified strategy for data integrity does not detect an SNMPv2 message modification which appends extraneous material to the end of such messages. However, owing to the representation of SNMPv2 messages as ASN.1 values, such modifications cannot - consistent with goal 1 - result in unauthorized management operations.

The data integrity mechanism specified in this memo protects only against unauthorized modification of individual SNMPv2 messages. A more general data integrity service that affords protection against the threat of message stream modification is not realized by this mechanism, although limited protection against reordering, delay, and duplication of messages within a message stream are provided by other mechanisms of the protocol.

A.3 Data Origin Authentication Mechanism

The data integrity mechanism requires the use of a secret value known only to communicating parties. By virtue of this mechanism and assumptions 1 and 2, the protocols explicitly prevent unauthorized modification of messages. Data origin authentication is implicit if the message digest value can be verified. That is, the protocols realize goal 2.

A.4 Data Restricted Authentication Mechanism

This memo requires that implementations preclude administrative alterations of the authentication clock for a particular party independently from its private authentication key (unless that clock alteration is an advancement).

A.5 Message Timeliness Mechanism

The definition of the SNMPv2 security protocols requires that, if the authentication timestamp value on a received message - augmented by an administratively chosen lifetime value - is less than the local notion of the clock for the source SNMPv2 party, the message is not delivered.

```
if (timestampOfReceivedMsg +
       party->administrativeLifetime <=
       party->localNotionOfClock) {
       msgIsValidated = FALSE;
}
```

By virtue of this mechanism, the protocols realize goal 3. In cases in which the local notions of a particular SNMPv2 party clock are moderately well-synchronized, the timeliness mechanism effectively limits the age of validly delivered messages. Thus, if an attacker diverts all validated messages for replay much later, the delay introduced by this attack is limited to a period that is proportional to the skew among local notions of the party clock.

A.6 Selective Clock Acceleration Mechanism

The definition of the SNMPv2 security protocols requires that, if either of the timestamp values for the source or destination parties on a received, validated message exceeds the corresponding local notion of the clock for that party, then the local notion of the clock for that party is adjusted forward to correspond to said timestamp value. This mechanism is neither strictly necessary nor sufficient to the security of the protocol; rather, it fosters the clock synchronization on which valid message delivery depends - thereby enhancing the effectiveness of the protocol in a management context.

```
if (msgIsValidated) {
       if (timestampOfReceivedMsg >
             party->localNotionOfClock) {
             party->localNotionOfClock =
                   timestampOfReceivedMsg;
       }
}
```

The effect of this mechanism is to synchronize local notions of a party clock more closely in the case where a sender's notion is more advanced than a receiver's. In the opposite case, this mechanism has no effect on local notions of a party clock and either the received message is validly delivered or not according to other mechanisms of the protocol.

Operation of this mechanism does not, in general, improve the probability of validated delivery for messages generated by party participants whose local notion of the party clock is relatively less advanced. In this case, queries from a management station may not be validly delivered and the management station needs to react appropriately (e.g., by use of the strategy described in Section 5.3). In contrast, the delivery of SNMPv2 trap messages generated by an agent that suffers from a less advanced notion of a party clock is more problematic, for an agent may lack the capacity to recognize and react to security failures that prevent delivery of its messages. Thus, the inherently unreliable character of trap messages is likely to be compounded by attempts to provide for their validated delivery.

A.7 Confidentiality Mechanism

The protocols require the use of a symmetric encryption algorithm when the data confidentiality service is required. By virtue of this mechanism and assumption 3, the protocols realize goal 4.

Authors' Addresses

Jeffrey D. Case +
SNMP Research, Inc. +
3001 Kimberlin Heights Rd. +
Knoxville, TN 37920-9716 +
US +

Phone: +1 615 573 1434 +
Email: case@snmp.com +

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

Phone: +1 301 854-6889
EMail: galvin@tis.com

Keith McCloghrie
Cisco Systems, Inc.
170 West Tasman Drive,
San Jose CA 95134-1706.

Phone: +1 408 526 5260
Email: kzm@cisco.com

Marshall T. Rose +
Dover Beach Consulting, Inc. +
420 Whisman Court +
Mountain View, CA 94043-2186 +
US +

Phone: +1 415 968 1052 +
Email: mrose@dbc.mtview.ca.us +

Steven Waldbusser +
Carnegie Mellon University +
5000 Forbes Ave +
Pittsburgh, PA 15213 +
US +

Phone: +1 412 268 6628 +
Email: waldbusser@cmu.edu +

Table of Contents