

On the Multicasting Security of SNMPv3

Salah A. Aly

Computer Science Dept., Texas A&M University, USA
salah@cs.tamu.edu

Hani Abu-Salem

Computer Science Dept., CTI, DePaul University, USA
habusalem@cs.depaul.edu

Mohamed Taha M. Abu-Kreisha

Faculty of Science, Cairo University, Egypt
mtahacom@hotmail.com

Abstract

SNMPv3, a simple network management protocol, is secure in unicasting communications (i.e., point-to-point or end-to-end transmission). Furthermore, a manager and an agent can authenticate and exchange secure transmission between each other with no time delay or replay. In this paper, we extend the SNMPv3 unicasting security solution to be applicable in multicasting applications. In addition, we present a security framework for multicasting authentication, privacy, and messages time delay and reply.

Keywords: *SNMPv3, network management, network security, multicasting.*

1 Introduction

SNMPv3 is one of the most widely-used network management protocols that is typically based on a connection-less transport service, which may operate over any sub-network service. The re-ordering, delaying, or replaying of messages occur through the natural operation of many such sub-network services. Message stream modification threat is the danger that messages may be maliciously re-ordered, delayed, or replayed to an extent. Many techniques for the unicasting security of SNMP have been proposed recently. For example, sending and receiving one message from a manager to an agent can be secured as Lam and Gouda mention in [1].

We aim in this paper to study the management security of SNMPv3 for multicasting usage. The idea of multicasting is that whenever a user (sender) sends messages, they go to a number of other users (receivers). We are not concerned about the messages themselves; however, we are concerned about end-to-end communications and end to multi-end transmissions. The routers and other network devices are responsible for the connection between the different network topologies. In SNMPv3 assumptions, there are two main entities, manager and group of agents. The manager can establish a secure channel with every member (agent), then forward one session key for all agents. A group session, on the other hand, may persist for a relatively long time with its members.

The requirements that a group key management protocol must satisfy are: scalability to large number of users, independence, reliability, security, and efficiency to support fast packet rates [2], [3], [4].

2 SNMP Architecture and Overview

SNMPv3 defines new MIB's that can be used for configuring security, notifications, proxy forwarding, and view-based access control. The SNMP MIB, a structured collection of all the managed objects maintained by a device, can be accessed remotely and by different users. Therefore, the user can control which MIB variables different users can have access to [5].

One of the most important features of SNMPv3 compared to the previous versions is its security mechanism. SNMPv3 covers this feature by offering both a strong authentication and privacy. Currently, SNMP uses HMAC-MD5-96 and HMAC-SHA-96 as authentication protocols and CBC-DES as a privacy protocol. In addition, SNMP defines a strong mechanism to protect messages against time delay and replay. It uses both the engine boot and time for detecting the timeliness of a message as shown in following sections.

SNMP engine security consists of two subsystems: User-based Security Model (USM) and Access Control Model (ACM). In this work, we focus on USM and how to manage its security services [6], [7], [8]. There are four basic messages that can be used in SNMP security:

- Get Message: a message sent from a manager to an agent.
- Set Message: a message sent from an agent to a manager.
- Trap Message: a message sent from an agent to a manager.
- Inform Request Message: a message sent from a manager to another manager.

The User-based Security Model (USM) simple discovery process has been defined to allow the non-authoritative entity (manager) to learn the SNMPEngineID of the authoritative entity (agent) that the non-authoritative entity might communicate with. It attempts to be secured against the following principal threats: disclosure, masquerading, modification, and loose message packets. But USM does not attempt to solve the following two problems: traffic analysis and denial of service.

An SNMP agent is a trusted object that can be added or removed from a multicasting group easily by the manager of this group. An SNMP agent is a non-authoritative object that can send trap messages to the manager. Examples of agents are PC's, workstations, servers, printers, etc. Properties of the agent are given in detail in Stalling's paper [6]. The manager is responsible for creating the keys, establishing the group, and managing the agents in this group. A traditional SNMP manager communicates with its SNMP agents by get and set messages, and by receiving a trap message [7], [8].

2.1 SNMPv3 Security

Based on the ongoing threats in the SNMP network management environment, the goals of this SNMP security model are as follows:

1. *Verify that each received SNMP message has not been modified during its transmission through the network.*
2. *Verify identity of the user on whose behalf a received SNMP message claims to have been generated.*

3. *Verify the detection of received SNMP messages, which request or contain management information, whose time of generation was not recent.*
4. *Verify, when necessary, that the content of each received SNMP message are protected from disclosure.*

The security services required to support the goals of this SNMP Security Model are as follows [6], [7], [8]:

- 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 identity of the user who the received data was created for is corroborated.
- Data confidentiality is the provision of the property that information is not made available or disclosed to unauthorized individuals, entities, or processes.
- Message timeliness and limited replay protection are the provision of the property that a message whose generation time is outside of a specified time window is not accepted.

2.2 Message Delay and Replay Protections [Timeliness Messages]

The exchanged messages must be protected against delay and replay attacks. Each SNMP engine maintains three objects:

1. `snmpEngineID`, which (at least within an administrative domain) uniquely and unambiguously identifies an SNMP engine.
2. `snmpEngineBoot`, which is a count of the number of times the SNMP engine has re-booted/re-initialized since `snmpEngineID` was last configured.
3. `snmpEngineTime`, which is the number of seconds since the `snmpEngineBoot` counter was last incremented.

The messages will be accepted if the `snmpEngineBoot` matches the non-authoritative known value, and the `snmpEngineTime` is within the defined `TimeWindow`. All SNMP engines are by default authoritative regarding their names. On the other hand, it is the responsibility of a non-authoritative SNMP engine to synchronize with the authoritative SNMP engine.

3 A Proposed Solution for Multicasting Usage

Using SNMPv3 Multicasting is important for developing scalable distributed network management applications. This reduces the monitoring latency and the exchanged SNMP messages significantly. This work proposes a flexible, efficient and easy-to-integrate framework for integrating IP Multicast in SNMPv3 agents. In this section, we discuss the extensions required to consider SNMPv3 security issues. Our suggested solution depends mainly on the group keys between the manager and agents of the same group.

As SNMPv3 supposes that there is a manager that creates new group keys. When establishing a group, the multicasting group keys can be sent via unicast to the members (encrypted with their individual keys) or assigned to them physically. The manager, after establishing the group, is responsible for updating multicasting keys, and evicting or adding members to this group.

Definition:

SNMPv3 multicasting privacy and authentication are a tuple $(M, A, \text{PrivKey}_g, \text{AuthKey}_g)$, where:

- I. M is a nonempty set of managers (non-authoritatives)
- II. A is a nonempty set of agents (authoritatives)
- III. PrivKey_g is a nonempty set of privacy keys (PrivKeys)

$$\text{for every } \text{PrivKey}_{gi} = f(M_i, \sum_{j=0}^n A_j)$$

- IV. AuthKey_g is a nonempty set of variant authentication keys (AuthKeys)

$$\text{where } \text{AuthKey}_{gi} = g(M_i, \sum_{j=0}^n A_j)$$

We suggest using the X-OR, \oplus or a secure Hash function for simplicity of computations for generating both PrivKey_g and AuthKey_g as in next section. However, PrivKey_g and AuthKey_g are still more secure since the concatenation between all local keys for the agents and the manager is used.

$$\begin{aligned} \text{PrivKey}_g &= \text{Hash}(\sum \text{EngineID's}, \text{MgrPassWord} \parallel \text{randomValue}) \\ \text{AuthKey}_g &= \text{Hash}(\text{PrivKey}_g, \text{MgrPassWord} \parallel \text{randomValue}) \end{aligned}$$

So, the attacker has to know the manager password and all engines ID's in order to create PrivKey_g . Also, it needs to know the manager password and PrivKey_g in order to gain AuthKey_g .

SNMP key values should be changed periodically; daily, weekly, or monthly. We propose a method where the key values can be updated. Simply, whenever the user changes his password, the key values will change automatically. If the random value in the origination of the key is changed, the PrivKey_g and AuthKey_g will be altered directly:

$$\begin{aligned} \text{PrivKey}_{g'} &= \text{Hash}(\text{PrivKey}_g, \text{MgrPassWord} \parallel \text{randomValue}) \\ \text{AuthKey}_{g'} &= \text{Hash}(\text{PrivKey}_{g'}, \text{MgrPassWord} \parallel \text{randomValue}) \end{aligned}$$

4 Privacy, Authentication and Messages Delay and Reply Mechanisms

In this section, we present multicasting security mechanisms for SNMPv3 for privacy, authentication, and messages time delay and reply.

4.1 SNMPv3 Multicasting Privacy Mechanism

Fig.1 illustrates the CDC-DES encrypted message for the multicasting usage. In general, the manager and the agents share a PrivKey_g for the group. However, the PrivKey_g is assigned by the manager for each agent allowed to have this key and access the data in the group.

The Initialization Vector (IV) for encryption is obtained using the following procedure. The last 8 octets of the 16-octet secret (PrivKey_g) are used as preIV. In order to ensure that the IV for two different packets encrypted by the same key, are not the same (i.e., the IV does not repeat) we need to "salt" the preIV with something unique per packet. An 8-

octet string is used as the "salt". The concatenation of the generating SNMP engine's 32-bit snmpGRoupID and a local 32-bit integer, that the encryption engine maintains, is input to the "salt". The 2-bit integer is initialized to an arbitrary value at boot time. The 32-bit snmpGRoupID is converted to the first 4 octets (Most Significant Byte first) of our "salt". The 32-bit integer is then converted to the last 4 octet (Most Significant Byte first) of our "salt". The resulting "salt" is then XOR-ed with the preIV to obtain the IV. The 8-octet "salt" is then put into the privParameters field encoded as an OCTET STRING. The "salt" integer is then modified. We recommend that it be incremented by one and wrap when it reaches the maximum value. How much the value of the "salt" (and thus of the IV) varies is an implementation issue, as long as measures are taken to avoid producing a duplicate IV. The "salt" must be placed in the privParameters field to enable the receiving entity to compute the correct IV and decrypt the message.

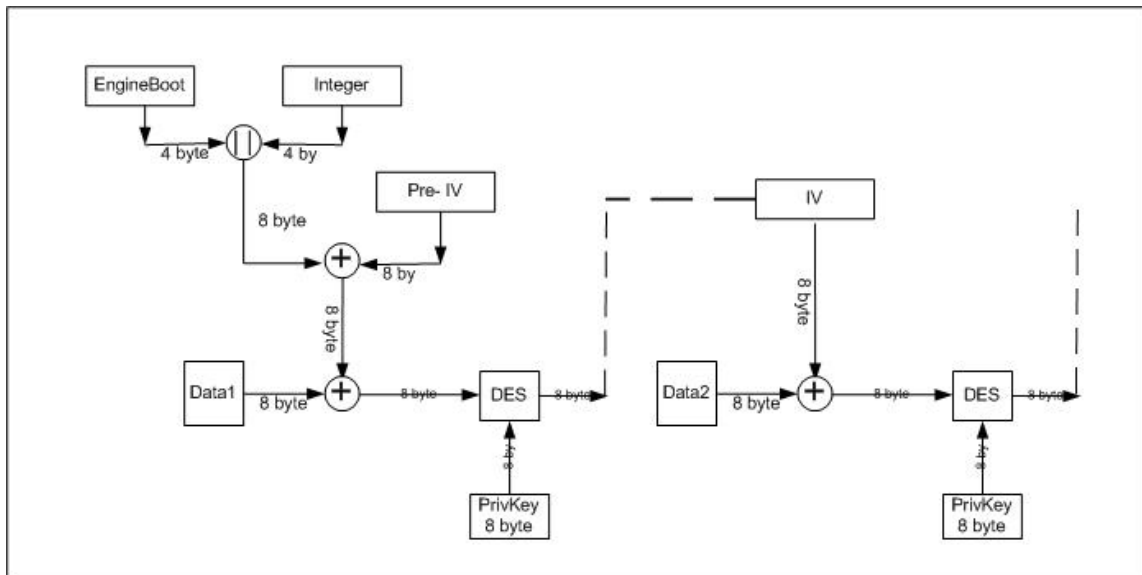


Figure 1: Encrypting the outgoing message using CBC-DES

4.1.1 Data Encryption

Fig.1 shows how to encrypt data using CBC-DES algorithm. The data to be encrypted is treated as sequence of octets. Its length should be an integral multiple of 8, and if it is not, the data is padded at the end as necessary. The actual pad value is irrelevant. The data is encrypted in Cipher Block Chaining mode (CBC-mode). The plaintext is divided into 64-bit blocks. The plaintext for each block is XOR-ed with the ciphertext of the previous block, the result is encrypted and the output of the encryption is the ciphertext for the block. This procedure is repeated until there are no more plaintext blocks. For the very first block, the Initialization Vector is used instead of the ciphertext of the previous block. The manager computes encrypted message as follows:

- Generate: $PrivKey_g = \text{Hash}(\sum (\text{EngineID's}), \text{MgrPassWord} \parallel \text{randomValue})$
- Compute: $IV = ((\text{snmpGRoupID} \parallel \text{LocalInteger}) \oplus \text{preIV})$
- Encrypt: $\text{EncryptData} = \text{DES}(Privkey_g, \text{Data} \oplus IV)$
- Send: EncryptData using User-base Security Model.

4.1.2 Data Decryption

In the decryption process, the first ciphertext block is decrypted, the decryption output is XOR-ed with the Initialization Vector, and the result is the first plaintext block. For each subsequent block, the ciphertext block is decrypted, the decryption output is XOR-ed with the previous ciphertext block and the result is the plaintext block. The agent operations are as follows:

- Receive: the key P rivKey and the preIV by using secure key exchange algorithm.
- Decrypt: DecryptData = DES(Privkey_g, EncryptData)
- Compute: IV = ((snmpGRoupID||LocalInteger) ⊕ preIV)
- Compute: Data = (DecryptData ⊕ IV)

4.2 SNMPv3 Multicasting Authentication Mechanism

As in the unicasting authentication, we can use HMAC-MD5-96 and HMAC-SHA-96 to ensure the data integrity between the multicasting groups.

Fig.2 illustrates HMAC-MD5 message authentication for the multicasting usage. The main difference between the SNMP authentication in unicasting and multicasting is the way the authentication key (AuthKey_g) is generated. For authentication using HMAC-SHA-96 the messages can be used as in the unicasting. Fig.2 illustrates HMAC-MD5 message authentication for the multicasting usage. The same technique can be used exactly for the HMAC-SHA-96 but we have to use SHA instead of MD5. Also, the key size in SHA is 20 octets instead of 16 octets as in MD5 authentication.

4.3 SNMPv3 Multicasting Messages Delay and Replay

As Al-Shaer states in [10], time synchronization is required by the non-authoritative entity and the agents in order to proceed with the authentication procedure. It occurs whenever the non-authoritative entity obtains new values for snmpEngineBoot and snmpEngineTime from an authoritative entity. It also occurs when the other agents receive a message from the non-authoritative or other agents in the same group. If the message received by the non-authoritative entity contains boot or time values that are greater than its local notion values, this means that the agent's parameters were changed and the local configurations of the non-authoritative must be updated accordingly in order to maintain the clock synchronization [8]. Now, we present some definitions of delay and reply between agents and a manager.

EngineBoot is the value of authoritative agent boot saved in the non-authoritative manager. **EngineTime** is the value of authoritative agent time saved in the non-authoritative manager. **EngineDeltaBoot** is the difference between the AgentBoot and the EngineBoot saved in the non-authoritative manager. **EngineDeltaTime** is the difference between the AgentTime and the EngineTime saved in the non-authoritative manager. **AgentDeltaBoot** is the difference between two agents boot in the same group saved in the agent. **AgentDeltaTime** is the difference between two agents time in the same group saved in the agent.

The relationships between these values are as follows:

$$\begin{aligned} \text{AgentDeltaBoot} &= \text{OldAgentDeltaBoot} + \text{EngineDeltaBoot} \\ \text{AgentDeltaTime} &= \text{OldAgentDeltaTime} + \text{EngineDeltaTime} \end{aligned}$$

$$\begin{aligned} \text{EngineDeltaBoot} &= \text{AgentBoot} - \text{EngineBoot} \\ \text{EngineDeltaTime} &= \text{AgentTime} - \text{EngineTime} \end{aligned}$$

When an agent reboots, it decrements value of AgentDeltaBoot and set value of AgentDeltaTime to zero.

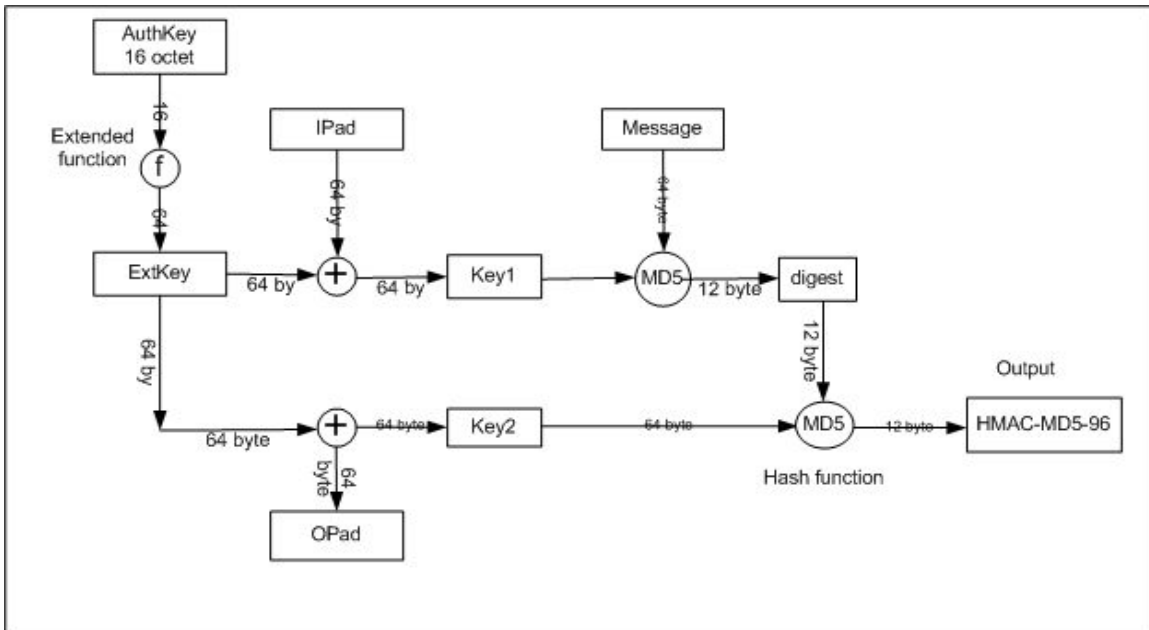


Figure 2: Generating multicasting message authentication code (MAC) using MD5

4.3.1 An authoritative agent timeliness and synchronization

We present a model for protecting messages against delaying and replaying when the authoritative agent receives multicasting messages. We suppose that values of the EngineBoot and EngineTime are equal to values of the AgentBoot and AgentTime only for the first authoritative agent. Moreover, all other agents consider the first agent as standard clock to adjust their deltas.

- The agents update their Deltas when they receive a message if the following cases occur:
 - if $\text{AgentBoot} < 2147483647$ and
 - if $\text{EngineBoot} = \text{AgentBoot} + \text{AgentDeltaBoot}$ and
 - if $|\text{EngineTime} - \text{AgentTime}| < 150$ then
$$\begin{aligned} \text{AgentDeltaBoot} &= \text{OldAgentDeltaBoot} + \text{EngineDeltaBoot} \\ \text{AgentDeltaTime} &= \text{OldAgentDeltaTime} + \text{EngineDeltaTime} \end{aligned}$$
- The agents discard a message if only one of the following cases is true:
 - $\text{AgentBoot} = 2147483647$ or
 - $\text{AgentBoot} + \text{AgentDeltaBoot} < \text{EngineBoot}$ or
 - $\text{AgentTime} + \text{AgentDeltaTime} > \text{EngineTime} \pm 150$

4.3.2 A non-authoritative manager timeliness and synchronization

- The non-authoritative manager updates** the value of the EngineBoot, EngineTime, EngineDeltaBoot, and EngineDeltaTime when it receives a message only from the first agent if one of the cases occur:

- EngineBoot = 2147483647 or
- AgentBoot < EngineBoot or
- AgentBoot = EngineBoot but AgentTime > EngineTime ± 150

The manager also discards any message from the other agents if one of the following cases is true:

- EngineBoot = 2147483647 or
- AgentBoot + AgentDeltaBoot <> EngineBoot or
- AgentBoot + AgentDeltaBoot = EngineBoot but
- AgentTime + AgentDeltaTime > EngineTime ± 150

Conclusion

We formalized a multicasting security framework for SNMPv3 and illustrated how the privacy and the authentication services can be achieved. We also demonstrated a technique for key generation between all agents in a multicasting group. We extended the unicasting security solution for the multicasting usage. Therefore, our multicasting security framework of SNMPv3 is simple and flexible for real applications.

References

- [1] Chung Kei Wong , Mohamed Gouda, and Simon S. Lam , Secure group communications using key graphs, IEEE/ACM Transactions on Networking, February 2000.
- [2] Suvo Mitra, Iolus: A framework for scalable secure multicasting in proceedings of ACM SIGCOMM '97, Cannes, France, September 1997.
- [3] Ehab Al-Shaer, Yongning Tang, Toward integrating IP multicasting in Internet network management protocols, Journal of Computer and Communications Review, December 2000.
- [4] Saswati Sarkar, Leandros Tassioulas, Back pressure based multicast scheduling for fair bandwidth allocation, Proceedings of INFOCOM 2001, Alaska, 2001.
- [5] David Zeltserman, A practical guide to SNMPv3 and network management, Prentice Hall, 1999.
- [6] Willam Stallings, SNMPv3: a security enhancement for SNMP, IEEE Communications Surveys, Vol.1 No.1, Fourth Quarter 1998.
- [7] Bert Wijnen, Uri Blumenthal, User-based security model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3), STD 62, RFC3414, December 2002.
- [8] Bert Wijnen, Uri Blumenthal, User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3), RFC2574, April 1999.
- [9] Harrington, D., Presuhn, R. and B. Wijnen, An architecture for describing Simple Network Management Protocol (SNMP) management Frameworks, STD 62, RFC3411, December 2002.
- [10] Ehab Al-Shaer, A framework for integrating IP multicasting in SNMPv3, Internet draft, September 2000.
- [11] Larry Korba, Security exposures with simple network management protocol, national research council of Canada, January 1999.
- [12] Peter S. Kruus, A survey of multicast security issues and architectures, NIST, 1998.
- [13] Adrian Perrig, Ron Canetti, Dong Song and J.D. Tygar, Efficient and secure source authentication for multicast, in Network and Distributed System Security Symposium, NDSS01 (2001).
- [14] SNMP v1, v2, v3 Research International, Inc 2004, FAQ, implementation, research products, and protocols of SNMPv3. <http://www.snmp.com/snmpv3/>