Network Working Group
Request for Comments: 3826
Category: Standards Track

U. Blumenthal
Lucent Technologies
F. Maino
Andiamo Systems, Inc.
K. McCloghrie
Cisco Systems, Inc.

June 2004

The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model

Status of this Memo

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

Copyright Notice

Abstract

This document describes a symmetric encryption protocol that supplements the protocols described in the User-based Security Model (USM), which is a Security Subsystem for version 3 of the Simple Network Management Protocol for use in the SNMP Architecture. The symmetric encryption protocol described in this document is based on the Advanced Encryption Standard (AES) cipher algorithm used in Cipher FeedBack Mode (CFB), with a key size of 128 bits.

Table of Contents

1. Introduction

Within the Architecture for describing Internet Management Frameworks [RFC3411], the User-based Security Model (USM) [RFC3414] for SNMPv3 is defined as a Security Subsystem within an SNMP engine. RFC 3414 describes the use of HMAC-MD5-96 and HMAC-SHA-96 as the initial authentication protocols, and the use of CBC-DES as the initial privacy protocol. The User-based Security Model, however, allows for other such protocols to be used instead of, or concurrently with, these protocols.

This memo describes the use of CFB128-AES-128 as an alternative privacy protocol for the User-based Security Model. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.1. Goals and Constraints

The main goal of this memo is to provide a new privacy protocol for the USM based on the Advanced Encryption Standard (AES) [FIPS-AES].

The major constraint is to maintain a complete interchangeability of the new protocol defined in this memo with existing authentication and privacy protocols already defined in USM.

For a given user, the AES-based privacy protocol MUST be used with one of the authentication protocols defined in RFC 3414 or an algorithm/protocol providing equivalent functionality.

1.2. Key Localization

As defined in [RFC3414], a localized key is a secret key shared between a user U and one authoritative SNMP engine E. Even though a user may have only one pair of authentication and privacy passwords (and consequently only one pair of keys) for the entire network, the actual secrets shared between the user and each authoritative SNMP engine will be different. This is achieved by key localization.

If the authentication protocol defined for a user U at the authoritative SNMP engine E is one of the authentication protocols defined in [RFC3414], the key localization is performed according to the two-step process described in section 2.6 of [RFC3414].

1.3. Password Entropy and Storage

The security of various cryptographic functions lies both in the strength of the functions themselves against various forms of attack, and also, perhaps more importantly, in the keying material that is used with them. While theoretical attacks against cryptographic functions are possible, it is more probable that key guessing is the main threat.

The following are recommended in regard to user passwords:

- Password length SHOULD be at least 12 octets.
- Password sharing SHOULD be prohibited so that passwords are not shared among multiple SNMP users.
- Implementations SHOULD support the use of randomly generated passwords as a stronger form of security.

It is worth remembering that, as specified in [RFC3414], if a user’s password or a non-localized key is disclosed, then key localization will not help and network security may be compromised. Therefore, a user’s password or non-localized key MUST NOT be stored on a managed device/node. Instead, the localized key SHALL be stored (if at all) so that, in case a device does get compromised, no other managed or managing devices get compromised.

2. Definitions

This MIB is written in SMIv2 [RFC2578].

```
SNMP-USM-AES-MIB DEFINITIONS ::= BEGIN
    IMPORTS
        MODULE-IDENTITY, OBJECT-IDENTITY,
        snmpModules             FROM SNMPv2-SMI           -- [RFC2578]
        snmpPrivProtocols       FROM SNMP-FRAMEWORK-MIB; -- [RFC3411]

snmpUsmAesMIB  MODULE-IDENTITY
    LAST-UPDATED "200406140000Z"
    ORGANIZATION "IETF"
    CONTACT-INFO "Uri Blumenthal
                  Lucent Technologies / Bell Labs
                  67 Whippany Rd.
                  14D-318
                  Whippany, NJ  07981, USA
                  973-386-2163
                  uri@bell-labs.com

                  Fabio Maino
                  Andiamo Systems, Inc.
                  375 East Tasman Drive
                  San Jose, CA  95134, USA
                  408-853-7530
                  fmaino@andiamo.com

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

                  408-526-5260
                  kzm@cisco.com"
    DESCRIPTION  "Definitions of Object Identities needed for
                  the use of AES by SNMP's User-based Security
                  Model.

                  Copyright (C) The Internet Society (2004).

            This version of this MIB module is part of RFC 3826;
            see the RFC itself for full legal notices.
            Supplementary information may be available on
            http://www.ietf.org/copyrights/ianamib.html."
    REVISION     "200406140000Z"
    DESCRIPTION  "Initial version, published as RFC3826"

    ::= { snmpModules 20 }

usmAesCfb128Protocol OBJECT-IDENTITY
    STATUS        current
    DESCRIPTION  "The CFB128-AES-128 Privacy Protocol."
    REFERENCE    "- Specification for the ADVANCED ENCRYPTION
                    STANDARD. Federal Information Processing
                    Standard (FIPS) Publication 197.
                    (November 2001).

                  - Dworkin, M., NIST Recommendation for Block
                    Cipher Modes of Operation, Methods and
                    Techniques. NIST Special Publication 800-38A
                    (December 2001).
                 "
    ::= { snmpPrivProtocols 4 }

END
```

3. CFB128-AES-128 Symmetric Encryption Protocol

This section describes a Symmetric Encryption Protocol based on the AES cipher algorithm [FIPS-AES], used in Cipher Feedback Mode as described in [AES-MODE], using encryption keys with a size of 128 bits.

This protocol is identified by usmAesCfb128PrivProtocol.

The protocol usmAesCfb128PrivProtocol is an alternative to the privacy protocol defined in [RFC3414].

3.1. Mechanisms

In support of data confidentiality, an encryption algorithm is required. An appropriate portion of the message is encrypted prior to being transmitted. The User-based Security Model specifies that the scopedPDU is the portion of the message that needs to be encrypted.

A secret value is shared by all SNMP engines which can legitimately originate messages on behalf of the appropriate user. This secret value, in combination with a timeliness value and a 64-bit integer, is used to create the (localized) en/decryption key and the initialization vector.

3.1.1. The AES-based Symmetric Encryption Protocol

The Symmetric Encryption Protocol defined in this memo provides support for data confidentiality. The designated portion of an SNMP message is encrypted and included as part of the message sent to the recipient.

The AES (Advanced Encryption Standard) is the symmetric cipher algorithm that the NIST (National Institute of Standards and Technology) has selected in a four-year competitive process as Replacement for DES (Data Encryption Standard).

The AES homepage, http://www.nist.gov/aes, contains a wealth of information on AES including the Federal Information Processing Standard [FIPS-AES] that fully specifies the Advanced Encryption Standard.

The following subsections contain descriptions of the relevant characteristics of the AES ciphers used in the symmetric encryption protocol described in this memo.

3.1.1.1. Mode of operation

The NIST Special Publication 800-38A [AES-MODE] recommends five confidentiality modes of operation for use with AES: Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR).

The symmetric encryption protocol described in this memo uses AES in CFB mode with the parameter S (number of bits fed back) set to 128 according to the definition of CFB mode given in [AES-MODE]. This mode requires an Initialization Vector (IV) that is the same size as the block size of the cipher algorithm.

3.1.1.2. Key Size

In the encryption protocol described by this memo AES is used with a key size of 128 bits, as recommended in [AES-MODE].

3.1.1.3. Block Size and Padding

The block size of the AES cipher algorithms used in the encryption protocol described by this memo is 128 bits, as recommended in [AES-MODE].

3.1.1.4. Rounds

This parameter determines how many times a block is encrypted. The encryption protocol described in this memo uses 10 rounds, as recommended in [AES-MODE].

3.1.2. Localized Key, AES Encryption Key, and Initialization Vector

The size of the Localized Key (Kul) of an SNMP user, as described in [RFC3414], depends on the authentication protocol defined for that user U at the authoritative SNMP engine E.

The encryption protocol defined in this memo MUST be used with an authentication protocol that generates a localized key with at least 128 bits. The authentication protocols described in [RFC3414] satisfy this requirement.

3.1.2.1. AES Encryption Key and IV

The first 128 bits of the localized key Kul are used as the AES encryption key. The 128-bit IV is obtained as the concatenation of the authoritative SNMP engine’s 32-bit snmpEngineBoots, the SNMP engine’s 32-bit snmpEngineTime, and a local 64-bit integer. The 64-bit integer is initialized to a pseudo-random value at boot time.

The IV is concatenated as follows: the 32-bit snmpEngineBoots is converted to the first 4 octets (Most Significant Byte first), the 32-bit snmpEngineTime is converted to the subsequent 4 octets (Most Significant Byte first), and the 64-bit integer is then converted to the last 8 octets (Most Significant Byte first). The 64-bit integer is then put into the msgPrivacyParameters field encoded as an OCTET STRING of length 8 octets. The integer is then modified for the subsequent message. We recommend that it is incremented by one until it reaches its maximum value, at which time it is wrapped.

An implementation can use any method to vary the value of the local 64-bit integer, providing the chosen method never generates a duplicate IV for the same key.

A duplicated IV can result in the very unlikely event that multiple managers, communicating with a single authoritative engine, both accidentally select the same 64-bit integer within a second. The probability of such an event is very low, and does not significantly affect the robustness of the mechanisms proposed.

The 64-bit integer must be placed in the privParameters field to enable the receiving entity to compute the correct IV and to decrypt the message. This 64-bit value is called the "salt" in this document.

Note that the sender and receiver must use the same IV value, i.e., they must both use the same values of the individual components used to create the IV. In particular, both sender and receiver must use the values of snmpEngineBoots, snmpEngineTime, and the 64-bit integer which are contained in the relevant message (in the msgAuthoritativeEngineBoots, msgAuthoritativeEngineTime, and privParameters fields respectively).

3.1.3. Data Encryption

The data to be encrypted is treated as a sequence of octets.

The data is encrypted in Cipher Feedback mode with the parameter s set to 128 according to the definition of CFB mode given in Section 6.3 of [AES-MODE]. A clear diagram of the encryption and decryption process is given in Figure 3 of [AES-MODE].

The plaintext is divided into 128-bit blocks. The last block may have fewer than 128 bits, and no padding is required.

The first input block is the IV, and the forward cipher operation is applied to the IV to produce the first output block. The first ciphertext block is produced by exclusive-ORing the first plaintext block with the first output block. The ciphertext block is also used as the input block for the subsequent forward cipher operation.

The process is repeated with the successive input blocks until a ciphertext segment is produced from every plaintext segment.

The last ciphertext block is produced by exclusive-ORing the last plaintext segment of r bits (r is less than or equal to 128) with the segment of the r most significant bits of the last output block.

3.1.4. Data Decryption

In CFB decryption, the IV is the first input block, the first ciphertext is used for the second input block, the second ciphertext is used for the third input block, etc. The forward cipher function is applied to each input block to produce the output blocks. The output blocks are exclusive-ORed with the corresponding ciphertext blocks to recover the plaintext blocks.

The last ciphertext block (whose size r is less than or equal to 128) is exclusive-ORed with the segment of the r most significant bits of the last output block to recover the last plaintext block of r bits.

3.2. Elements of the AES Privacy Protocol

This section contains definitions required to realize the privacy modules defined by this memo.

3.2.1. Users

Data en/decryption using this Symmetric Encryption Protocol makes use of a defined set of userNames. For any user on whose behalf a message must be en/decrypted at a particular SNMP engine, that SNMP engine must have knowledge of that user. An SNMP engine that needs to communicate with another SNMP engine must also have knowledge of a user known to that SNMP engine, including knowledge of the applicable attributes of that user.

A user and its attributes are defined as follows:

<userName>
An octet string representing the name of the user.

<privAlg>
The algorithm used to protect messages generated on behalf of the user from disclosure.

<privKey>
The user's secret key to be used as input to the generation of the localized key for encrypting/decrypting messages generated on behalf of the user. The length of this key MUST be greater than or equal to 128 bits (16 octets).

<authAlg>
The algorithm used to authenticate messages generated on behalf of the user, which is also used to generate the localized version of the secret key.

3.2.2. msgAuthoritativeEngineID

The msgAuthoritativeEngineID value contained in an authenticated message specifies the authoritative SNMP engine for that particular message (see the definition of SnmpEngineID in the SNMP Architecture document [RFC3411]).

The user's (private) privacy key is different at each authoritative SNMP engine, and so the snmpEngineID is used to select the proper key for the en/decryption process.

3.2.3. SNMP Messages Using this Privacy Protocol

Messages using this privacy protocol carry a msgPrivacyParameters field as part of the msgSecurityParameters. For this protocol, the privParameters field is the serialized OCTET STRING representing the "salt" that was used to create the IV.

3.2.4. Services provided by the AES Privacy Modules

This section describes the inputs and outputs that the AES Privacy module expects and produces when the User-based Security module invokes one of the AES Privacy modules for services.

3.2.4.1. Services for Encrypting Outgoing Data

The AES privacy protocol assumes that the selection of the privKey is done by the caller, and that the caller passes the localized secret key to be used.

Upon completion, the privacy module returns statusInformation and, if the encryption process was successful, the encryptedPDU and the msgPrivacyParameters encoded as an OCTET STRING. The abstract service primitive is:

```
statusInformation =              -- success or failure
  encryptData(
  IN    encryptKey               -- secret key for encryption
  IN    dataToEncrypt            -- data to encrypt (scopedPDU)
  OUT   encryptedData            -- encrypted data (encryptedPDU)
  OUT   privParameters           -- filled in by service provider
        )
```

The abstract data elements are:

statusInformation
An indication of the success or failure of the encryption process. In case of failure, it is an indication of the error.

encryptKey
The secret key to be used by the encryption algorithm. The length of this key MUST be 16 octets.

dataToEncrypt
The data that must be encrypted.

encryptedData
The encrypted data upon successful completion.

privParameters
The privParameters encoded as an OCTET STRING.

3.2.4.2. Services for Decrypting Incoming Data

This AES privacy protocol assumes that the selection of the privKey is done by the caller and that the caller passes the localized secret key to be used.

Upon completion the privacy module returns statusInformation and, if the decryption process was successful, the scopedPDU in plain text. The abstract service primitive is:

```
statusInformation =
  decryptData(
  IN    decryptKey               -- secret key for decryption
  IN    privParameters           -- as received on the wire
  IN    encryptedData            -- encrypted data (encryptedPDU)
  OUT   decryptedData            -- decrypted data (scopedPDU)
        )
```

The abstract data elements are:

statusInformation
An indication of whether the data was successfully decrypted, and if not, an indication of the error.

decryptKey
The secret key to be used by the decryption algorithm. The length of this key MUST be 16 octets.

privParameters
The 64-bit integer to be used to calculate the IV.

encryptedData
The data to be decrypted.

decryptedData
The decrypted data.

3.3. Elements of Procedure

This section describes the procedures for the AES privacy protocol for SNMP's User-based Security Model.

3.3.1. Processing an Outgoing Message

This section describes the procedure followed by an SNMP engine whenever it must encrypt part of an outgoing message using the usmAesCfb128PrivProtocol.

1) The secret encryptKey is used to construct the AES encryption key, as described in section 3.1.2.1.

2) The privParameters field is set to the serialization according to the rules in [RFC3417] of an OCTET STRING representing the 64-bit integer that will be used in the IV as described in section 3.1.2.1.

3) The scopedPDU is encrypted (as described in section 3.1.3) and the encrypted data is serialized according to the rules in [RFC3417] as an OCTET STRING.

4) The serialized OCTET STRING representing the encrypted scopedPDU together with the privParameters and statusInformation indicating success is returned to the calling module.

3.3.2. Processing an Incoming Message

This section describes the procedure followed by an SNMP engine whenever it must decrypt part of an incoming message using the usmAesCfb128PrivProtocol.

1) If the privParameters field is not an 8-octet OCTET STRING, then an error indication (decryptionError) is returned to the calling module.

2) The 64-bit integer is extracted from the privParameters field.

3) The secret decryptKey and the 64-bit integer are then used to construct the AES decryption key and the IV that is computed as described in section 3.1.2.1.

4) The encryptedPDU is then decrypted (as described in section 3.1.4).

5) If the encryptedPDU cannot be decrypted, then an error indication (decryptionError) is returned to the calling module.

6) The decrypted scopedPDU and statusInformation indicating success are returned to the calling module.

4. Security Considerations

The security of the cryptographic functions defined in this document lies both in the strength of the functions themselves against various forms of attack, and also, perhaps more importantly, in the keying material that is used with them. The recommendations in Section 1.3 SHOULD be followed to ensure maximum entropy to the selected passwords, and to protect the passwords while stored.

The security of the CFB mode relies upon the use of a unique IV for each message encrypted with the same key [CRYPTO-B]. If the IV is not unique, a cryptanalyst can recover the corresponding plaintext.

Section 3.1.2.1 defines a procedure to derive the IV from a local 64-bit integer (the salt) initialized to a pseudo-random value at boot time. An implementation can use any method to vary the value of the local 64-bit integer, providing the chosen method never generates a duplicate IV for the same key.

The procedure of section 3.1.2.1 suggests a method to vary the local 64-bit integer value that generates unique IVs for every message. This method can result in a duplicated IV in the very unlikely event that multiple managers, communicating with a single authoritative engine, both accidentally select the same 64-bit integer within a second. The probability of such an event is very low, and does not significantly affect the robustness of the mechanisms proposed.

This AES-based privacy protocol MUST be used with one of the authentication protocols defined in RFC 3414 or with an algorithm/protocol providing equivalent functionality (including integrity), because CFB encryption mode does not detect ciphertext modifications.

For further security considerations, the reader is encouraged to read [RFC3414], and the documents that describe the actual cipher algorithms.

5. IANA Considerations

IANA has assigned OID 20 for the snmpUsmAesMIB module under the snmpModules subtree, maintained in the registry at http://www.iana.org/assignments/smi-numbers.

IANA has assigned OID 4 for the usmAesCfb128Protocol under the snmpPrivProtocols registration point, as defined in RFC 3411 [RFC3411].

6. Acknowledgements

Portions of this text, as well as its general structure, were unabashedly lifted from [RFC3414]. The authors are grateful to many of the SNMPv3 WG members for their help, especially Wes Hardaker, Steve Moulton, Randy Presuhn, David Town, and Bert Wijnen. Security discussions with Steve Bellovin helped to streamline this protocol.

7. References

7.1. Normative References

[AES-MODE] Dworkin, M., "NIST Recommendation for Block Cipher Modes of Operation, Methods and Techniques", NIST Special Publication 800-38A, December 2001.

[FIPS-AES] "Specification for the ADVANCED ENCRYPTION STANDARD (AES)", Federal Information Processing Standard (FIPS) Publication 197, November 2001.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2578] McCloghrie, K., Perkins, D. and J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC3411] Harrington, D., Presuhn, R. and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.

[RFC3414] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.

[RFC3417] Presuhn, R., Ed., "Transport Mappings for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3417, December 2002.

7.2. Informative References

[CRYPTO-B] Bellovin, S., "Probable Plaintext Cryptanalysis of the IP Security Protocols", Proceedings of the Symposium on Network and Distributed System Security, San Diego, CA, pp. 155-160, February 1997.

8. Authors’ Addresses

Uri Blumenthal
Lucent Technologies / Bell Labs
67 Whippany Rd.
14D-318
Whippany, NJ 07981, USA

Phone: +1-973-386-2163
EMail: uri@bell-labs.com

Fabio Maino
Andiamo Systems, Inc.
375 East Tasman Drive
San Jose, CA. 95134 USA

Phone: +1-408-853-7530
EMail: fmaino@andiamo.com

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

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

9. Full Copyright Statement

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.