

**Extension to the User-Based Security Model (USM) to  
Support Triple-DES EDE in "Outside" CBC Mode  
<[draft-reeder-snmpv3-usm-3desede-00.txt](#)>**

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC2026](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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.''

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

Abstract

This document describes an extension to the the User-based Security Model (USM) [[RFC2574](#)]. It defines the Elements of Procedure for providing SNMP message level security for a privacy protocol using Triple-DES EDE in "Outside" Cipher-Block Chaining (CBC) mode. This document also extends the existing SNMPv3 MIBs in order to remotely monitor and manage the configuration parameters for this privacy protocol of the USM.

## Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

## Table of Contents

1. Introduction	2
2. Use of Password-to-Key Algorithm with 3DES-EDE	5
2.1 Chaining of the Password-to-Key Algorithm	5
2.2 Password (P) versus Ku versus the localized key Kul	6
3. Use of SNMP-USER-BASED-SM-3DES-MIB MIB Module	7
4. Definitions	8
5. 3DES-EDE Symmetric Encryption Protocol	11
5.1. Mechanisms	11
5.1.1. Symmetric Encryption Protocol	11
5.1.1.1. 3DES-EDE Key and Initialization Vector	12
5.1.1.1.1. 3DES-EDE Key	12
5.1.1.1.2. 3DES-EDE Initialization Vector	13
5.1.1.2. Data Encryption	14
5.1.1.3. Data Decryption	15
5.2. Elements of the 3DES-EDE Privacy Protocol	16
5.2.1. Users	16
5.2.2. msgAuthoritativeEngineID	16
5.2.3. SNMP Messages Using this Privacy Protocol	17
5.2.4. Services provided by the 3DES-EDE Privacy Module	17
5.2.4.1. Services for Encrypting Outgoing Data	17
5.2.4.2. Services for Decrypting Incoming Data	18
5.3. Elements of Procedure	19
5.3.1. Processing an Outgoing Message	19
5.3.2. Processing an Incoming Message	19
6. Security Considerations	20
7. Acknowledgements	20
8. Intellectual Property	20
9. References	21
10. Editors' Addresses	23
11. Full Copyright Statement	24
A. SNMP engine Installation Parameters Using 3DES-EDE	24
B. Password-to-Key Chaining Sample Results	25
B.1. Password-to-Key Chaining Sample Results using MD5	25
B.2. Password-to-Key Chaining Sample Results using SHA	26
C. Sample keyChange Results	26
C.1. Sample keyChange Results using MD5	26
C.2. Sample keyChange Results using SHA	27
D.1. Strength of 3DES-EDE and Known Attacks	28

## D.2. Further References

29

**1. Introduction**

The Architecture for describing Internet Management Frameworks [[RFC2571](#)] describes that an SNMP engine is composed of:

- 1) a Dispatcher
- 2) a Message Processing Subsystem,
- 3) a Security Subsystem, and
- 4) an Access Control Subsystem.

Applications make use of the services of these subsystems.

It is important to understand the SNMP architecture and the terminology of the architecture to understand where the extensions to the Security Model described in this document fit into the architecture and interact with other subsystems within the architecture. The reader is expected to have read and understood the description of the SNMP architecture and the User-Based Security Model (USM), as defined in [[RFC2571](#)] and [[RFC2574](#)], respectively.

This memo describes an extension to the User-based Security Model which defines the 3DES-EDE privacy protocol using Triple-DES EDE in "Outside" CBC mode. "EDE" is one method of triple encryption which simply varies the "direction" that the single DES algorithm is used in each iteration -- first encrypting, then decrypting, and finally encrypting again. Since the second decryption uses a different key than the first encryption, the decryption iteration serves to further encrypt the data.

This extension adds to, but does not otherwise change, the details describing the use of privacy protocols in the USM. Further, it makes no changes to the remainder of the USM, and adopts all its assumptions, supporting concepts and apparatus. It is expected that this document alone will provide all necessary details necessary for the immediate integration and use of 3DES-EDE into the existing USM subsystem.

In particular, the following aspects of the USM are adopted in their entirety, without modification, by the 3DES-EDE extension:

- Abstract Service Interface describing the User-based Security Model primitives for privacy,
- Format of SNMP messages using the User-based Security Model,
- Password-to-key key localization algorithm,
- Services for generating an outgoing SNMP Message, and for processing an incoming SNMP message,
- Existing USM security considerations including:
  - \* Recommended practices
  - \* Defining users
  - \* Conformance
  - \* Use of reports
  - \* Access to the SNMP-USER-BASED-SM-MIB.

Note that some details surrounding the use of the password-to-key algorithm for key localization are necessarily changed when using this extension in order to provide for the larger number of bits required by the 3DES-EDE cryptographic key. The key localization algorithm as specified in the USM as the password-to-key algorithm has not changed, however. (See [[LOCALIZED-KEY](#)] for the original definition.)

In addition, while users may specify passphrases of any length, the maximum length of keying material used by the SNMP engine is limited to the length of the largest hash generated by the currently specified authentication protocols. Some effort should be taken to provide for key lengths greater than this protocols provide. Work in this regard, however, is outside the scope of this document.

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](#)].

## **2. Use of Password-to-Key Algorithm with 3DES-EDE**

A set of sample code fragments given in the USM document demonstrate the password-to-key algorithm which can be used to map a password to a privacy key using MD5 or SHA.

### **2.1 Chaining of the Password-to-Key Algorithm**

Some cryptographic algorithms may require keys that have a length greater than the that of the hash output used by the password-to-key algorithm. This will be the case, for example, with any user that defines `usm3DESEDEPrivProtocol` as its privacy protocol (described below in [Section 6](#)). To acquire the necessary number of key bits, the password-to-key algorithm may be chained using its own output as further input in order to generate an appropriate number of key bits.

Chaining is described as follows. First, run the password-to-key algorithm with inputs of the passphrase and `engineID` as described in the USM document. This will output as many key bits as the hash algorithm used to implement the password-to-key algorithm. Secondly, run the password-to-key algorithm again with the previous output (instead of the passphrase) and the same `engineID` as inputs. Repeat this process as many times as necessary in order to generate the minimum number of key bits for the chosen privacy protocol. The outputs of each execution are concatenated into a single string of key bits.

When this process results in more key bits than are necessary, only the most significant bits of the string should be used.

For example, if password-to-key implemented with SHA creates a 40-octet string for use as key bits, only the first 32 octets will be used for `usm3DESEDEPrivProtocol`.

Chaining may be demonstrated using simplified pseudo-code as follows, let:

```
Output_bits <-- P2K( Input_bits, EngineID )
```

where the string of key bits (`Output_bits`) is returned from the password-to-key (`P2K`) algorithm which takes a string of bits (`Input_bits`) and the `engineID` (`EngineID`) as inputs. One iteration of chaining, creating a localized key of twice the normal length is achieved as follows:

```
K1 <-- P2K( <passphrase>, <engine_id> )
K2 <-- P2K( K1,           <engine_id> )

localized_key = K1 | K2
```

The next further iteration will pass K2 (instead of K1) and return K3. The iteration after that passes K3 and returns K4, etc. The results of all iterations (K1, K2, ..., Kn) are concatenated to form the localized key. Note that the engineID is the same for all iterations.

An example of the results of a correct implementation is provided in [Appendix B](#).

## **2.2 Password (P) versus Ku versus the localized key Kul**

It is important to note that using the chaining method confuses the simple relationship between the passphrase, Ku and the localized key, Kul described in the USM document. It is believed that this should pose no significant difficulty to for existing USM implementations, however.

The password-to-key algorithm performs two actions. First, it derives Ku from P by expanding P to 1024 kilobytes and hashing the result. Second, it derives Kul from Ku by hashing a concatenation of Ku and engineID. This latter step constitutes the localization method.

Normally, Ku is temporarily generated within the password-to-key algorithm only for use in generating Kul, and it is expected that Ku will be discarded after this use. When the password-to-key algorithm is chained as described in [Section 2.1](#), the final string of key bits output is no longer directly derived from P through Ku. Further there is no longer a one-to-one mapping between P and Ku, and from Ku to Kul. Nonetheless, the cryptographic mixing and unifying function provided by chaining the password-to-key algorithm serves the same purpose as a single use of the password-to-key algorithm.

Alternatives to the chaining method might require the password-to-key algorithm to take an input indicating the number of key bits desired, allowing the algorithm to perform the entire chaining operation (or some other pseudo-random number generation technique).

The benefits of chaining the password-to-key algorithm, effectively using it "as is," include the following:

- Should be simpler for existing implementations to adapt to the longer 3DES-EDE key length with a minimum of changes.
- No need to document and define a choice between the existing password-to-key algorithm and some new algorithm.

The drawbacks of the described chaining method include:

- The notion of  $K_u$  and its relationship to  $P$  and  $K_{u1}$  is confused.
- Network management stations that insist on storing  $K_u$  will have to store the passphrase ( $P$ ) instead.

Note that storing  $P$  poses the same security risks as storing  $K_u$ .

- A new algorithm could be optimized to save at least one hashing operation per chaining cycle.

### **3. Use of SNMP-USER-BASED-SM-3DES-MIB MIB Module**

The current purpose of the SNMP-USER-BASED-SM-3DES-MIB MIB module is simply to define the OBJECT-IDENTITY `usm3DESEDEPrivProtocol`. This adds to but does not change the { `snmpModules snmpFrameworkMIB` } MIB module [[RFC2571](#)] by naming a new privacy protocol.

This naming takes place within the context of the USM { `snmpModules snmpUsmMIB` } MIB module where other privacy protocols have previously been defined. When the 3DES-EDE privacy protocol is used, a size of `SIZE(0 | 64)` is recommended for use with the following OBJECT-TYPES:

```
{ usmUserEntry usmUserPrivKeyChange }  
{ usmUserEntry usmUserOwnPrivKeyChange }.
```

#### [4.](#) **Definitions**



SNMP-USER-BASED-SM-3DES-MIB DEFINITIONS ::= BEGIN

IMPORTS

MODULE-IDENTITY, OBJECT-IDENTITY,  
    snmpModules                            FROM SNMPv2-SMI  
AutonomousType                            FROM SNMPv2-TC  
snmpPrivProtocols                        FROM SNMP-FRAMEWORK-MIB;

snmpUsmMIB MODULE-IDENTITY

LAST-UPDATED "9910060000Z"                -- 06 October 1999, midnight

ORGANIZATION "SNMPv3 Working Group"

CONTACT-INFO "WG-email: snmpv3@lists.tislabs.com

Subscribe: majordomo@lists.tislabs.com

In msg body: subscribe snmpv3

Chair: Russ Mundy

NAI Labs

postal: 3060 Washington Rd  
Glenwood MD 21738  
USA

email: mundy@tislabs.com

phone: +1-443-259-2307

Co-editor: David Reeder

NAI Labs

postal: 3060 Washington Road (Route 97)  
Glenwood, MD 21738  
USA

email: dreeder@tislabs.com

phone: +1-443-259-2348

Co-editor: Olafur Gudmundsson

NAI Labs

postal: 3060 Washington Road (Route 97)  
Glenwood, MD 21738  
USA

email: ogud@tislabs.com

phone: +1-443-259-2389

"

DESCRIPTION "Extension to the SNMP User-based Security Model  
to support Triple-DES EDE in 'Outside' CBC  
(cipher-block chaining) Mode.

"

-- Revision history

REVISION "9910060000Z" -- 06 October 1999, midnight  
DESCRIPTION "Initial version, published as an Internet Draft."

::= { snmpModules 15 }

-- Identification of Privacy Protocols \*\*\*\*\*

-- Note: { snmpPrivProtocols 1 } through { snmpPrivProtocols 2 }  
-- are defined in USM.

usm3DESEDEPrivProtocol OBJECT-IDENTITY

STATUS current

DESCRIPTION "The 3DES-EDE Symmetric Encryption Protocol."

REFERENCE "- Data Encryption Standard, National Institute of  
Standards and Technology. Federal Information  
Processing Standard (FIPS) Publication 46-3,

(1999,

pending approval). Will supersede FIPS

Publication

46-2.

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

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

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

"

::= { snmpPrivProtocols 3 }

END

## **5. 3DES-EDE Symmetric Encryption Protocol**

This section describes the 3DES-EDE Symmetric Encryption Protocol. This protocol is an optional privacy protocol defined for use with the User-based Security Model.

This protocol is identified by `usm3DESEDEPrivProtocol`.

Over time, other privacy protocols may be defined either as a replacement of this protocol or in addition to this protocol.

### **5.1. 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 in combination with a timeliness value is used to create the en/decryption key and the initialization vector. The secret value is shared by all SNMP engines authorized to originate messages on behalf of the appropriate user.

#### **5.1.1. Symmetric Encryption Protocol**

The Symmetric Encryption Protocol defined here 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.

Two organizations have published specifications defining 3DES-EDE: the National Institute of Standards and Technology (NIST) [[3DES-NIST](#)] and the American National Standards Institute [[3DES-ANSI](#)]. There is a companion Modes of Operation specification for each definition ([[DESO-NIST](#)] and [[DESO-ANSI](#)], respectively). Additional information about 3DES-EDE may be found in [[SCHNEIER95](#)] (see Chapter 12 and [Section 15.2](#)).

The NIST has published three additional documents that implementors may find useful. Although these documents were written with (single) DES in mind, they may be adapted to the use of 3DES-EDE.

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

- There is a specification of a validation test suite for the DES [[DEST-NIST](#)]. 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 [[DESM-NIST](#)]. 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 re-boots.

#### **5.1.1.1.1. 3DES-EDE Key and Initialization Vector**

##### **5.1.1.1.1.1. 3DES-EDE Key**

The first 24 octets of the 32-octet secret (private privacy key) are used as a 3DES-EDE key. Since 3DES-EDE uses only 168 bits, the Least Significant Bit in each octet is disregarded.

The 3DES-EDE subkeys <K1, K2, K3> are obtained in the following manner.

The 24-octet sequence is divided into three smaller 8-octet sequences where bytes 1 through 8 define K1, bytes 9 through 16 define K2, and bytes 17 through 24 define K3. For each 8-octet sequence, bytes are assigned to its respective subkey from left to right, beginning with the most significant byte and extending to the least significant byte.

The three subkeys MUST be verified to be different. This may be done by checking that K1 is not equal to K2, and that K2 is not equal to K3. This will guarantee that at least two of the three subkeys are different. To verify that all three subkeys are different, it SHOULD be verified in addition that K1 is not equal to K3. The first set of checks verifies that the whole key contains at least 112 bits, the second check verifies that the whole key contains 168 bits. For a stronger key it is advised that both checks are performed.

There are no published attacks against 3DES-EDE that take advantage of using "weak keys" for any of K1, K2 or K3. The list of weak keys includes the semi-weak and possibly-weak keys. It is generally accepted that 3DES-EDE is resistant to attacks using these keys, unlike single DES. Nonetheless, since the list of weak keys is small, it is advised that each of the subkeys SHOULD be checked for membership in this list. The complete list of known weak keys is given in [[SCHNEIER95](#)].

The checks for difference and weakness noted above should be performed when the key is assigned. If any of the mandated tests fail, then the whole key MUST be discarded and an appropriate exception noted.

#### **5.1.1.1.2. 3DES-EDE Initialization Vector**

The Initialization Vector for encryption is obtained using the following procedure.

The last 8 octets of the 32-octet secret (private privacy key) are used as pre-IV.

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 over the lifetime of the private key) we need to "salt" the pre-IV with something unique per packet. An 8-octet string is used as the "salt". The concatenation of the generating SNMP engine's 32-bit snmpEngineBoots and a local 32-bit integer, that the encryption engine maintains, is input to the "salt". The 32-bit integer is initialized to an arbitrary value at boot time.

This 32-bit arbitrary integer SHOULD be randomly determined when the engine boots. This is in keeping with the recommendations given in [RFC1750] and [PLAIN-ANALYSIS] with regard to the generation of random numbers and the use of predictable plaintext to speed a cryptographic search for a secret key.

If the arbitrary integer cannot be chosen randomly, it is suggested instead that it SHOULD be derived from a hardware clock, or from system.sysUpTime.0 if a hardware clock is not available. These options are preferable to a simple counter as periodic use of it will not describe a direct sequence of natural numbers.

The 32-bit snmpEngineBoots 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 octets (Most Significant Byte first) of our "salt".

To achieve effective bit spreading, the complete 8-octet "salt" value SHOULD be hashed using the usmUserAuthProtocol. This may be performed using the authentication algorithm directly, or by passing the "salt" as input the the password-to-key algorithm. The result of the hash is truncated to 8 octets.

The resulting "salt" is then XOR-ed with the pre-IV to obtain the IV. The 8-octet "salt" is then put into the privParameters field encoded as an OCTET STRING.

Finally, the "salt" value is updated in preparation for future use, possibly using one of the methods just described. How exactly the value of the "salt" varies (and thus the value of the IV), is an implementation issue, as long as the measures are taken to avoid producing a duplicate IV over the lifetime of the private key.

The "salt" must be placed in the privParameters field to enable the receiving entity to compute the correct IV and to decrypt the message.

#### **5.1.1.2. Data Encryption**

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 using Triple DES Encryption - Decryption - Encryption (EDE) in Outside Cipher Block Chaining (CBC) mode.

The plaintext is divided into 64-bit blocks. A single block of plaintext is encrypted by performing a sequence of encryption with the first key (K1), followed by decryption of the previous result with the second key (K2), finally followed by encryption of the previous result with the final key (K3).

The plaintext for each block is XOR-ed with the ciphertext of the previous EDE encryption, the result is EDE 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 (IV) is used instead of the ciphertext of the previous block.

This is expressed more succinctly by the following formula:

$$C_i = E_{K3}( D_{K2}( E_{K1}( P_i \text{ XOR } C_{(i-1)} ) ) ) )$$

Where plaintext block number  $i$  is XOR-ed with ciphertext block number  $(i-1)$ , then encrypted with  $K1$ , decrypted with  $K2$ , and encrypted again with  $K3$  to give ciphertext block number  $i$ . For the first EDE

encryption,  $C(i-1)$  is replaced by the IV. A more thorough explanation may be found in [[SCHNEIER95](#)].

#### **5.1.1.3. Data Decryption**

Before decryption, the encrypted data length is verified. If the length of the OCTET STRING to be decrypted is not an integral multiple of 8 octets, the decryption process is halted and an appropriate exception noted. When decrypting, the padding is ignored.

The first ciphertext block is decrypted, the decryption output is XOR-ed with the Initialization Vector, and the result is the first plaintext block.

A single ciphertext block is decrypted by performing a sequence of decryption with the third key ( $K_3$ ), followed by encryption of the previous result with the second key ( $K_2$ ), finally followed by decryption of the previous result with the first key ( $K_1$ ). This cycle of decryption - encryption - decryption (DED) is the reverse of the EDE sequence used for encryption.

For each subsequent block, the ciphertext block is DED decrypted, then the decryption output is XOR-ed with the previous ciphertext block and the result is the plaintext block.

This is expressed more succinctly by the following formula:

$$P_i = C(i-1) \text{ XOR } D_{K1}( E_{K2}( D_{K3}( C_i ) ) )$$

Where ciphertext block number  $i$  is decrypted with  $K_3$ , then encrypted with  $K_2$ , then decrypted with  $K_1$  and finally XOR-ed with ciphertext block  $(i-1)$  to give plaintext block number  $i$ . For the first ciphertext block of the series,  $C(i-1)$  is replaced by the IV. A more thorough explanation may be found in [[SCHNEIER95](#)].

## **5.2. Elements of the 3DES-EDE Privacy Protocol**

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

### **5.2.1. 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 wishes 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.

<privKey>

A user's secret key to be used as input for the 3DES-EDE key and IV. The length of this key MUST be 32 octets.

### **5.2.2. 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 [[RFC2571](#)]).

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



### **5.2.3. SNMP Messages Using this Privacy Protocol**

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

### **5.2.4. Services provided by the 3DES-EDE Privacy Module**

This section describes the inputs and outputs that the 3DES-EDE Privacy module expects and produces when the User-based Security module invokes the 3DES-EDE Privacy module for services.

#### **5.2.4.1. Services for Encrypting Outgoing Data**

This 3DES-EDE privacy protocol assumes that the selection of the privKey is done by the caller and that the caller passes the 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 of 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 32 octets.
dataToEncrypt
  The data that must be encrypted.
encryptedData
  The encrypted data upon successful completion.
privParameters
```

The privParameters encoded as an OCTET STRING.

#### **5.2.4.2. Services for Decrypting Incoming Data**

This 3DES-EDE privacy protocol assumes that the selection of the privKey is done by the caller and that the caller passes the 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 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 32 octets.  
encryptedData  
  The data to be decrypted.  
decryptedData  
  The decrypted data.  
privParameters  
  The "salt" to be used to calculate the IV.
```

### **5.3. Elements of Procedure**

This section describes the procedures for the 3DES-EDE privacy protocol.

#### **5.3.1. 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 `usm3DESEDEPrivProtocol`.

- 1) The secret `cryptKey` is used to construct the 3DES-EDE encryption key, the "salt" and the 3DES-EDE pre-IV (from which the IV is computed as described in [section 5.1.1.1.2](#)).
- 2) The `privParameters` field is set to the serialization according to the rules in [[RFC1906](#)] of an OCTET STRING representing the the "salt" string.
- 3) The `scopedPDU` is encrypted (as described in [section 5.1.1.2](#)) and the encrypted data is serialized according to the rules in [[RFC1906](#)] 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.

#### **5.3.2. 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 `usm3DESEDEPrivProtocol`.

- 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 "salt" is extracted from the `privParameters` field.
- 3) The secret `cryptKey` and the "salt" are then used to construct the 3DES-EDE decryption key and pre-IV (from which the IV is computed as described in [section 5.1.1.1.2](#)).
- 4) The `encryptedPDU` is then decrypted (as described in [section 5.1.1.3](#)).

- 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.

## **6. Security Considerations**

This document fully adopts and expects enforcement of the details presented in the Security Considerations section of the document describing the User-based Security Model [[RFC2574](#)].

Insofar as the privacy protocol presented in this document is considered to be an improvement over existing SNMP privacy protocols, this document presents an alternative offering greater security for the SNMP architecture.

## **7. Acknowledgements**

The general structure of this document and some of the text in it was taken directly from the document describing the User-based Security Model. Many details and references specific to the strength and analysis of the Triple-DES cryptographic algorithm were initially adapted from the description of that algorithm given in documents generated by the IPsec Working Group concerning the Encapsulation Security Protocol [[ESP-DESCBC](#)][ESP-3DES].

## **8. Intellectual Property**

The IETF takes no position regarding the validity or scope of any intellectual property 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; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in [BCP-11](#). Copies of claims of rights made available for publication 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 implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

## 9. References

- [3DES-ANSI] Data Encryption Algorithm, American National Standards Institute. ANSI X9.52.
- [3DES-NIST] Data Encryption Standard, National Institute of Standards and Technology. Federal Information Processing Standard (FIPS) Publication 46-3, (1999, pending approval). Will supersede FIPS Publication 46-2. <http://csrc.nist.gov/fips/dfips46-3.pdf>  
<http://csrc.nist.gov/cryptval/des/fr990115.htm>
- [DESG-NIST] 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). <http://www.itl.nist.gov/fipspubs/fip74.htm>
- [DESO-ANSI] Data Encryption Algorithm - Modes of Operation, American National Standards Institute. ANSI X3.106- 1983, (May 1983).
- [DESO-NIST] DES Modes of Operation, National Institute of Standards and Technology. Federal Information Processing Standard (FIPS) Publication 81, (December, 1980).  
<http://www.itl.nist.gov/fipspubs/fip81.htm>
- [DESM-NIST] Maintenance Testing for the Data Encryption Standard, National Institute of Standards and Technology. Special Publication 500-61, (August, 1980).
- [DEST-NIST] Validating the Correctness of Hardware Implementations of the NBS Data Encryption Standard, National Institute of Standards and Technology. Special Publication 500-20.

- [ESP-DESCBC] Madson, C. and N. Dorswamy, "The ESP DES-CBC Cipher Algorithm With Explicit IV", [RFC 2405](#), November 1998.
- [ESP-3DES] Doraswamy, N., Metzger, P., and Simpson, W.A., "The ESP Triple DES Transform", <[draft-ietf-ipsec-ciph-des3-00.txt](#)>, July 1997.  
<http://www.ietf.org/internet-drafts/draft-ietf-ipsec-ciph-des3-00.txt>
- [IETF-CRYPTO] Schiller, J., "Cryptographic Algorithms for the IETF," <[draft-ietf-saag-aes-ciph-00.txt](#)>, August 1999. <http://web.mit.edu/network/sa/draft-ietf-saag-aes-ciph-00.txt>
- [LOCALIZED-KEY] U. Blumenthal, N. C. Hien, B. Wijnen "Key Derivation for Network Management Applications" IEEE Network Magazine, April/May issue, 1997.
- [MIN-KEYLENGTH] Blaze, M., Diffie, W., Rivest, R., Schneier, B., Shimomura, T., Thompson, E., and M. Wiener, "Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security", currently available at  
<http://www.crypto.com/papers/keylength.ps>  
<http://www.crypto.com/papers/keylength.txt>
- [PLAIN-ANALYSIS] 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.  
<http://www.research.att.com/~smb/papers/probtxt.ps>  
<http://www.research.att.com/~smb/papers/probtxt.pdf>
- [RFC1750] Eastlake, D., Crocker, S., and J. Schiller, "Randomness Recommendations for Security", [RFC 1750](#), December 1994.  
<http://www.ietf.org/rfc/rfc1750.txt>
- [RFC1906] Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", [RFC 1906](#), January 1996.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2571] Harrington, D., Presuhn, R. and B. Wijnen, "An Architecture for describing SNMP Management Frameworks", [RFC 2571](#), April 1999.
- [RFC2574] Blumenthal, U. and B. Wijnen, "The User-Based Security Model for Version 3 of the Simple Network Management Protocol (SNMPv3)", [RFC 2574](#), April 1999.
- [SCHNEIER95] Schneier, B., "Applied Cryptography Second Edition", John Wiley & Sons, New York, NY, 1995. ISBN 0-471-12845-7.

## **10. Editors' Addresses**

David Reeder  
NAI Labs  
3060 Washington Road (Route 97)  
Glenwood, MD 21738  
USA

Phone: +1-443-259-2348  
Email: dreeder@tislabs.com

Olafur Gudmundsson  
NAI Labs  
3060 Washington Road (Route 97)  
Glenwood, MD 21738  
USA

Phone: +1-443-259-2389  
Email: ogud@tislabs.com

## **11. Full Copyright Statement**

Copyright (C) The Internet Society (1999). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Appendix

### **A. SNMP engine Installation Parameters Using 3DES-EDE**

In order to use the 3DES-EDE privacy protocol in place of the CBC-DES, the SNMP engine may use the following usmUserEntry in the initial configuration of the usmUserTable:



	privacy support
	-----
usmUserEngineID	localEngineID
usmUserName	"initial"
usmUserSecurityName	"initial"
usmUserCloneFrom	ZeroDotZero
usmUserAuthProtocol	usmHMACMD5AuthProtocol
usmUserAuthKeyChange	""
usmUserOwnAuthKeyChange	""
usmUserPrivProtocol	usm3DESEDEPrivProtocol
usmUserPrivKeyChange	""
usmUserOwnPrivKeyChange	""
usmUserPublic	""
usmUserStorageType	anyValidStorageType
usmUserStatus	active

Templates instantiated as initial usmUserEntries for use as clone-from users have a similar format. The usmUserPrivProtocol of usm3DESEDEPrivProtocol replaces usmDESPrivProtocol.

## **B. Password-to-Key Chaining Sample Results**

### **B.1. Password-to-Key Chaining Sample Results using MD5**

[Please Note: This note will be removed when the following values have been double-checked by a third party.]

The following shows a sample output of the password-to-key algorithm for a 32-octet key using MD5. The password used in this example is "maplesyrup". The first 16 octets (bytes 1 through 16) are generated by password-to-key algorithm with the passphrase as input. The second 16 octets (bytes 17 through 32) are generated from the password-to-key algorithm with the first 16 octets as input.

Each invocation of the password-to-key algorithm in the generation of a string of key bits uses the same engineID. In this example the engineID is:

```
'00 00 00 00 00 00 00 00 00 00 00 02'H
```

The final output of the password-to-key algorithm, used twice as described above, produces a 32-octet localized key of:

```
'52 6f 5e ed 9f cc e2 6f 89 64 c2 93 07 87 d8 2b
79 ef f4 4a 90 65 0e e0 a3 a4 0a bf ac 5a cc 12'H
```

## **B.2. Password-to-Key Chaining Sample Results using SHA**

[Please Note: This note will be removed when the following values have been double-checked by a third party.]

The following shows a sample output of the password-to-key algorithm for a 40-octet key using SHA. The password used in this example is "maplesyrup". The first 20 octets (bytes 1 through 20) are generated by password-to-key algorithm with the passphrase as input. The second 20 octets (bytes 21 through 40) are generated from the password-to-key algorithm with the first 20 octets as input.

Each invocation of the password-to-key algorithm in the generation of a string of key bits uses the same engineID. In this example the engineID is:

```
'00 00 00 00 00 00 00 00 00 00 00 00 02'H
```

The final output of the password-to-key algorithm, used twice as described above, produces a 40-octet localized key of:

```
'66 95 fe bc 92 88 e3 62 82 23 5f c7 15 1f 12 84 97 b3 8f 3f
9b 8b 6d 78 93 6b a6 e7 d1 9d fd 9c d2 d5 06 55 47 74 3f b5'H
```

## **C. Sample keyChange Results for 32-octet keys**

### **C.1. Sample keyChange Results for 32-octet Keys Using MD5**

[Please Note: This section is incomplete.]

Let us assume that a user has a current password of "maplesyrup" as in section B.1. and let us also assume the snmpEngineID of 12 octets:

```
'00 00 00 00 00 00 00 00 00 00 00 00 02'H
```

If we now want to change the password to "newsyrup", then we first calculate the localized key for the new password. It is as follows:

```
87 02 1d 7b d9 d1 01 ba 05 ea 6e 3b f9 d9 bd 4a
70 29 8b 75 7c 91 99 b6 a8 fb f3 93 7b e0 54 XX'H
```

Then, using the following value as a placeholder for the random value:

```
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'H
```

we compute a keyChange value of:

```
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX'H
```

### **C.2. Sample keyChange Results for 32-octet Keys Using SHA**

[Please Note: This section is incomplete.]

Let us assume that a user has a current password of "maplesyrup" as in section B.2. and let us also assume the snmpEngineID of 12 octets:

```
'00 00 00 00 00 00 00 00 00 00 00 00 02'H
```

If we now want to change the password to "newsyrup", then we first calculate the localized key for the new password. It is as follows:

```
78 e2 dc ce 79 d5 94 03 b5 8c 1b ba a5 bf f4 63
91 f1 cd 25 97 74 35 55 f9 fc f9 4a c3 e7 e9 22'H
```

Note that this value has been truncated from to 32 octets.

Then, using the following value as a placeholder for the random value:

```
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'H
```

we compute a keyChange value of:

```
'00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX'H
```

### **D.1. Strength of 3DES-EDE and Known Attacks**

Although 3DES-EDE has an effective key length of 168 bits ( $56 * 3$ ), it may be attacked with brute force as though its key were only 112 bits via the meet-in-the-middle attack [[MULTI-CRYPT](#)]. Even so, this number of key bits is greater than the minimum currently recommended by expert cryptanalysts, although it is still somewhat short of the most conservative estimates [[MIN-KEYLENGTH](#)].

It has been demonstrated that a DES key may be recovered by differential cryptanalysis [[DIFF-ANALYSIS](#)] and linear cryptanalysis [[LIN-ANALYSIS](#)] after collecting a minimum of  $2^{47}$  and  $2^{43}$  plaintext/ciphertext pairs, respectively. 3DES-EDE is susceptible to the same attacks given the same number of plaintext/ciphertext pairs [[DESMODES](#)].

Thus the primary value of 3DES-EDE over DES is not so much that it is more resistant to published theoretical attacks, but that it is apparently more resistant to brute force attacks.

[[DIFF-ANALYSIS](#)] also demonstrates a rare attack which requires only  $2^{33}$  plaintext/ciphertext pairs. For this reason, it is recommended that keys are changed after no more than  $2^{32}$  block encryptions.

Finally, as has been demonstrated in the context of IP Security, it is often a simpler and highly successful technique to guess at the contents of an encrypted block, and use these guesses in combination with differential or linear cryptanalysis to increase the probability of recovering the secret key [[PLAIN-ANALYSIS](#)]. SNMP has possible vulnerabilities in this regard as the following PDU fields are likely to be easily predictable by a passive observer:

- PDU Type
- Request ID
- Error Status, Error Index
- Non-Repeaters, Max-Repetition

Implementations may be classified by the species of their ASN.1 encoding engines, just as network hosts and routers may be classified by the species of their TCP/IP stack. This in combination with knowledge of common PDU exchanges makes the prediction of PDU fields a realistic endeavor.

Suggestions in [[PLAIN-ANALYSIS](#)] for closing these sorts of security holes include:

- \* Starting counters at random values,
- \* Replacing predictable values with random values when they are already known by the receiver,
- \* Keyed compression.

Concerns of this nature, however, are beyond the scope of this document.

## **D.2. Further References**

- [DESMODES] Biham, E., "On Modes of Operation," Fast Software Encryption, Cambridge Security Workshop Proceedings, Springer-Verlag, 1994, pp. 116-120.
- [DIFF-ANALYSIS] Biham, E., and Shamir, A., "Differential Cryptanalysis of the Data Encryption Standard", Berlin: Springer-Verlag, 1993.
- [LIN-ANALYSIS] Matsui, M., "Linear Cryptanalysis method for DES Cipher," Advances in Cryptology -- Eurocrypt '93 Proceedings, Berlin: Springer-Verlag, 1994.
- [MULTI-CRYPT] Merkle, R.C., and Hellman, M., "On the Security of Multiple Encryption", Communications of the ACM, v. 24 n. 7, 1981, pp. 465-467.