             Microsoft Point-To-Point Encryption (MPPE) Protocol

Status of this Memo

Copyright Notice

Abstract

   The Point-to-Point Protocol (PPP) provides a standard method for
   transporting multi-protocol datagrams over point-to-point links.

   The PPP Compression Control Protocol provides a method to negotiate
   and utilize compression protocols over PPP encapsulated links.

   This document describes the use of the Microsoft Point to Point
   Encryption (MPPE) to enhance the confidentiality of PPP-encapsulated
   packets.

Specification of Requirements

   In this document, the key words "MAY", "MUST, "MUST NOT", "optional",
   "recommended", "SHOULD", and "SHOULD NOT" are to be interpreted as
   described in [5].

1. Introduction

   The Microsoft Point to Point Encryption scheme is a means of
   representing Point to Point Protocol (PPP) packets in an encrypted
   form.

   MPPE uses the RSA RC4 [3] algorithm to provide data confidentiality.
   The length of the session key to be used for initializing encryption
   tables can be negotiated. MPPE currently supports 40-bit and 128-bit
   session keys.

MPPE session keys are changed frequently; the exact frequency depends upon the options negotiated, but may be every packet.
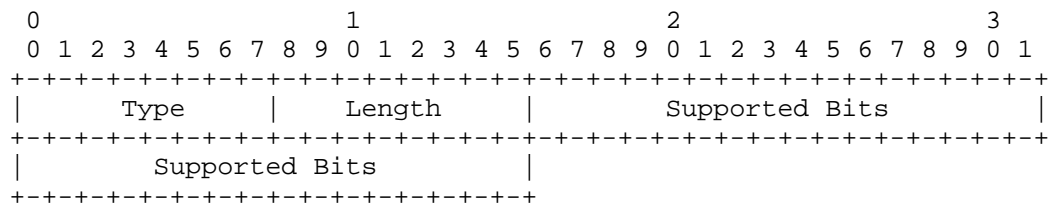
MPPE is negotiated within option 18 [4] in the Compression Control Protocol.

2. Configuration Option Format

   Description

      The CCP Configuration Option negotiates the use of MPPE on the
      link. By default (i.e., if the negotiation of MPPE is not
      attempted), no encryption is used. If, however, MPPE negotiation
      is attempted and fails, the link SHOULD be terminated.

   A summary of the CCP Configuration Option format is shown below. The
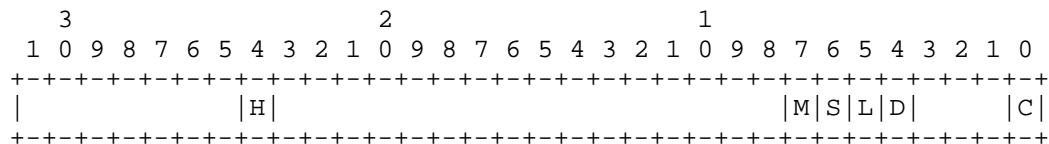   fields are transmitted from left to right.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Type      |    Length     |         Supported Bits        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Supported Bits        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Type

      18

   Length

      6

   Supported Bits

      This field is 4 octets, most significant octet first.

```
      3                   2                   1
      1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |             |H|                             |M|S|L|D|     |C|
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The 'C' bit is used by MPPC [4] and is not discussed further in this
memo. The 'D' bit is obsolete; although some older peers may attempt
to negotiate this option, it SHOULD NOT be accepted. If the 'L' bit
is set (corresponding to a value of 0x20 in the least significant
octet), this indicates the desire of the sender to negotiate the use
of 40-bit session keys. If the 'S' bit is set (corresponding to a
value of 0x40 in the least significant octet), this indicates the
desire of the sender to negotiate the use of 128-bit session keys.
If the 'M' bit is set (corresponding to a value of 0x80 in the least
significant octet), this indicates the desire of the sender to
negotiate the use of 56-bit session keys. If the 'H' bit is set
(corresponding to a value of 0x01 in the most significant octet),
this indicates that the sender wishes to negotiate the use of
stateless mode, in which the session key is changed after the
transmission of each packet (see section 10, below). In the
following discussion, the 'S', 'M' and 'L' bits are sometimes
referred to collectively as "encryption options".

All other bits are reserved and MUST be set to 0.

2.1. Option Negotiation

MPPE options are negotiated as described in [2]. In particular, the
negotiation initiator SHOULD request all of the options it supports.
The responder SHOULD NAK with a single encryption option (note that
stateless mode may always be negotiated, independent of and in
addition to an encryption option). If the responder supports more
than one encryption option in the set requested by the initiator, the
option selected SHOULD be the "strongest" option offered.
Informally, the strength of the MPPE encryption options may be
characterized as follows:

        STRONGEST
           128-bit encryption ('S' bit set)
           56-bit encryption ('M' bit set)
           40-bit encryption ('L' bit set)
        WEAKEST

This characterization takes into account the generally accepted
strength of the cipher.

The initiator SHOULD then either send another request containing the
same option(s) as the responder's NAK or cancel the negotiation,
dropping the connection.

3. MPPE Packets

     Before any MPPE packets are transmitted, PPP MUST reach the Network-
     Layer Protocol phase and the CCP Control Protocol MUST reach the
     Opened state.

     Exactly one MPPE datagram is encapsulated in the PPP Information
     field. The PPP Protocol field indicates type 0x00FD for all
     encrypted datagrams.

     The maximum length of the MPPE datagram transmitted over a PPP link
     is the same as the maximum length of the Information field of a PPP
     encapsulated packet.

     Only packets with PPP Protocol numbers in the range 0x0021 to 0x00FA
     are encrypted. Other packets are not passed thru the MPPE processor
     and are sent with their original PPP Protocol numbers.

        Padding

           It is recommended that padding not be used with MPPE. If the
           sender uses padding it MUST negotiate the Self-Describing-
           Padding Configuration option [10] during LCP phase and use
           self-describing pads.

        Reliability and Sequencing

           The MPPE scheme does not require a reliable link. Instead, it
           relies on a 12-bit coherency count in each packet to keep the
           encryption tables synchronized. If stateless mode has not been
           negotiated and the coherency count in the received packet does
           not match the expected count, the receiver MUST send a CCP
           Reset-Request packet to cause the resynchronization of the RC4
           tables.

           MPPE expects packets to be delivered in sequence.

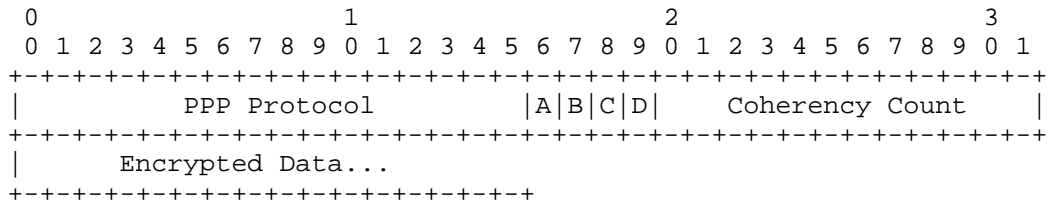   MPPE MAY be used over a reliable link, as described in "PPP
           Reliable Transmission" [6], but this typically just adds
           unnecessary overhead since only the coherency count is
           required.

        Data Expansion

           The MPPE scheme does not expand or compress data. The number
           of octets input to and output from the MPPE processor are the
           same.

3.1. Packet Format

   A summary of the MPPE packet format is shown below. The fields are
   transmitted from left to right.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            PPP Protocol        |A|B|C|D|   Coherency Count     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Encrypted Data...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   PPP Protocol

      The PPP Protocol field is described in the Point-to-Point
      Protocol Encapsulation [1].

      When MPPE is successfully negotiated by the PPP Compression
      Control Protocol, the value of this field is 0x00FD. This
      value MAY be compressed when Protocol-Field-Compression is
      negotiated.

   Bit A

      This bit indicates that the encryption tables were initialized
      before this packet was generated. The receiver MUST re-
      initialize its tables with the current session key before
      decrypting this packet. This bit is referred to as the FLUSHED
      bit in this document. If the stateless option has been
      negotiated, this bit MUST be set on every encrypted packet.
      Note that MPPC and MPPE both recognize the FLUSHED bit;
      therefore, if the stateless option is negotiated, it applies to
      both MPPC and MPPE.

   Bit B

      This bit does not have any significance in MPPE.

   Bit C

      This bit does not have any significance in MPPE.

   Bit D

      This bit set to 1 indicates that the packet is encrypted. This
      bit set to 0 means that this packet is not encrypted.

Coherency Count

The coherency count is used to assure that the packets are sent
in proper order and that no packet has been dropped. It is a
monotonically increasing counter which incremented by 1 for
each packet sent. When the counter reaches 4095 (0x0FFF), it
is reset to 0.

Encrypted Data

The encrypted data begins with the protocol field. For
example, in case of an IP packet (0x0021 followed by an IP
header), the MPPE processor will first encrypt the protocol
field and then encrypt the IP header.

If the packet contains header compression, the MPPE processor
is applied AFTER header compression is performed and MUST be
applied to the compressed header as well. For example, if a
packet contained the protocol type 0x002D (for a compressed
TCP/IP header), the MPPE processor would first encrypt 0x002D
and then it would encrypt the compressed Van-Jacobsen TCP/IP
header.

Implementation Note

If both MPPE and MPPC are negotiated on the same link, the MPPE
processor MUST be invoked after the MPPC processor by the
sender and the MPPE processor MUST be invoked before the MPPC
processor by the receiver.

4. Initial Session Keys

In the current implementation, initial session keys are derived from
peer credentials; however, other derivation methods are possible.
For example, some authentication methods (such as Kerberos [8] and
TLS [9]) produce session keys as side effects of authentication;
these keys may be used by MPPE in the future. For this reason, the
techniques used to derive initial MPPE session keys are described in
separate documents.

5. Initializing RC4 Using a Session Key

Once an initial session key has been derived, the RC4 context is
initialized as follows:

    rc4_key(RC4Key, Length_Of_Key, Initial_Session_Key)

6. Encrypting Data

   Once initialized, data is encrypted using the following function and
   transmitted with the CCP and MPPE headers.

      EncryptedData = rc4(RC4Key, Length_Of_Data, Data)

7. Changing Keys

7.1. Stateless Mode Key Changes

   If stateless encryption has been negotiated, the session key changes
   every time the coherency count changes; i.e., on every packet. In
   stateless mode, the sender MUST change its key before encrypting and
   transmitting each packet and the receiver MUST change its key after
   receiving, but before decrypting, each packet (see "Synchronization",
   below).

7.2. Stateful Mode Key Changes

   If stateful encryption has been negotiated, the sender MUST change
   its key before encrypting and transmitting any packet in which the
   low order octet of the coherency count equals 0xFF (the "flag"
   packet), and the receiver MUST change its key after receiving, but
   before decrypting, a "flag" packet (see "Synchronization", below).

7.3. The MPPE Key Change Algorithm

   The following method is used to change keys:

```
    /*
     * SessionKeyLength is 8 for 40-bit keys, 16 for 128-bit keys.
     *
     * SessionKey is the same as StartKey in the first call for
     * a given session.
     */

    void
    GetNewKeyFromSHA(
    IN  unsigned char *StartKey,
    IN  unsigned char *SessionKey,
    IN  unsigned long SessionKeyLength
    OUT unsigned char *InterimKey )
    {
       unsigned char  Digest[20];

       ZeroMemory(Digest, 20);
```

```
        /*
         * SHAInit(), SHAUpdate() and SHAFinal()
         * are an implementation of the Secure
         * Hash Algorithm [7]
         */

        SHAInit(Context);
        SHAUpdate(Context, StartKey, SessionKeyLength);
        SHAUpdate(Context, SHApad1, 40);
        SHAUpdate(Context, SessionKey, SessionKeyLength);
        SHAUpdate(Context, SHApad2, 40);
        SHAFinal(Context, Digest);

        MoveMemory(InterimKey, Digest, SessionKeyLength);
    }
```

The RC4 tables are re-initialized using the newly created interim key:

```
    rc4_key(RC4Key, Length_Of_Key, InterimKey)
```

Finally, the interim key is encrypted using the new tables to produce
a new session key:

```
    SessionKey = rc4(RC4Key, Length_Of_Key, InterimKey)
```

For 40-bit session keys the most significant three octets of the new
session key are now set to 0xD1, 0x26 and 0x9E respectively; for 56-
bit keys, the most significant octet is set to 0xD1.

Finally, the RC4 tables are re-initialized using the new session key:

```
    rc4_key(RC4Key, Length_Of_Key, SessionKey)
```

8. Synchronization

    Packets may be lost during transfer. The following sections describe
    synchronization for both the stateless and stateful cases.

8.1. Stateless Synchronization

    If stateless encryption has been negotiated and the coherency count
    in the received packet (C1) is greater than the coherency count in
    the last packet previously received (C2), the receiver MUST perform N
    = C1 - C2 key changes before decrypting the packet, in order to
    ensure that its session key is synchronized with the session key of
    the sender. Normally, the value of N will be 1; however, if
    intervening packets have been lost, N may be greater than 1. For
    example, if C1 = 5 and C2 = 02 then N = 3 key changes are required.

Since the FLUSHED bit is set on every packet if stateless encryption
was negotiated, the transmission of CCP Reset-Request packets is not
required for synchronization.

8.2. Stateful Synchronization

If stateful encryption has been negotiated, the sender MUST change
its key before encrypting and transmitting any packet in which the
low order octet of the coherency count equals 0xFF (the "flag"
packet), and the receiver MUST change its key after receiving, but
before decrypting, a "flag" packet. However, the "flag" packet may
be lost. If this happens, the low order octet of the coherency count
in the received packet will be less than that in the last packet
previously received. In this case, the receiver MUST perform a key
change before decrypting the newly received packet, (since the sender
will have changed its key before transmitting the packet), then send
a CCP Reset-Request packet (see below). It is possible that 256 or
more consecutive packets could be lost; the receiver SHOULD detect
this condition and perform the number of key changes necessary to
resynchronize with the sender.

If packet loss is detected while using stateful encryption, the
receiver MUST drop the packet and send a CCP Reset-Request packet
without data. After transmitting the CCP Reset-Request packet, the
receiver SHOULD silently discard all packets until a packet is
received with the FLUSHED bit set. On receiving a packet with the
FLUSHED bit set, the receiver MUST set its coherency count to the one
received in that packet and re-initialize its RC4 tables using the
current session key:

    rc4_key(RC4Key, Length_Of_Key, SessionKey)

When the sender receives a CCP Reset-Request packet, it MUST re-
initialize its own RC4 tables using the same method and set the
FLUSHED bit in the next packet sent. Thus synchronization is
achieved without a CCP Reset-Ack packet.

9. Security Considerations

Because of the way that the RC4 tables are reinitialized during
stateful synchronization, it is possible that two packets may be
encrypted using the same key. For this reason, the stateful mode
SHOULD NOT be used in lossy network environments (e.g., layer two
tunnels on the Internet).

   Since the MPPE negotiation is not integrity protected, an active
   attacker could alter the strength of the keys used by modifying the
   Supported Bits field of the CCP Configuration Option packet. The
   effects of this attack can be minimized through appropriate peer
   configuration, however.

   Peers MUST NOT transmit user data until the MPPE negotiation is
   complete.

   It is possible that an active attacker could modify the coherency
   count of a packet, causing the peers to lose synchronization.

   An active denial-of-service attack could be mounted by methodically
   inverting the value of the 'D' bit in the MPPE packet header.

10. References

   [1] Simpson, W., Editor, "The Point-to-Point Protocol (PPP)", STD
        51, RFC 1661, July 1994.

   [2] Rand, D., "The PPP Compression Control Protocol (CCP)", RFC
        1962, June 1996.

   [3] RC4 is a proprietary encryption algorithm available under
        license from RSA Data Security Inc. For licensing information,
        contact:

                RSA Data Security, Inc.
                100 Marine Parkway
                Redwood City, CA 94065-1031

   [4] Pall, G., "Microsoft Point-to-Point Compression (MPPC)
        Protocol", RFC 2118, March 1997.

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

   [6] Rand, D., "PPP Reliable Transmission", RFC 1663, July 1994.

   [7] "Secure Hash Standard", Federal Information Processing Standards
        Publication 180-1, National Institute of Standards and
        Technology, April 1995.

   [8] Kohl, J. and C. Neuman "The Kerberos Network Authentication
        System (V5)", RFC 1510, September 1993.

   [9] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC
        2246, January 1999.

   [10] Simpson, W., Editor, "PPP LCP Extensions", RFC 1570, January
        1994.

11. Acknowledgements

   Anthony Bell, Richard B. Ward, Terence Spies and Thomas Dimitri, all
   of Microsoft Corporation, significantly contributed to the design and
   development of MPPE.

   Additional thanks to Robert Friend, Joe Davies, Jody Terrill, Archie
   Cobbs, Mark Deuser, and Jeff Haag, for useful feedback.

12. Authors' Addresses

   Questions about this memo can be directed to:

   Gurdeep Singh Pall
   Microsoft Corporation
   One Microsoft Way
   Redmond, Washington 98052
   USA

   Phone: +1 425 882 8080
   Fax: +1 425 936 7329
   EMail: gurdeep@microsoft.com

   Glen Zorn
   cisco Systems
   500 108th Avenue N.E.
   Suite 500
   Bellevue, Washington 98004
   USA

   Phone: +1 425 438 8218
   Fax: +1 425 438 1848
   EMail: gwz@cisco.com

13. Full Copyright Statement

Acknowledgement