1. Status of this Memo

This document is an Internet-Draft. 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".

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

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. The distribution of this memo is unlimited. It is filed as <draft-ietf-pppext-mschapv1-keys-00.txt> and expires March 24, 1999. Please send comments to the PPP Extensions Working Group mailing list (ietf-ppp@merit.edu) or to the author (glennz@microsoft.com).

2. Abstract

The Point-to-Point Protocol (PPP) [1] provides a standard method for transporting multi-protocol datagrams over point-to-point links.


The Microsoft Challenge-Handshake Authentication Protocol (MS-CHAP) [3] is a Microsoft-proprietary PPP authentication protocol, providing the functionality to which LAN-based users are accustomed while integrating the encryption and hashing algorithms used on Windows networks.

Microsoft Point to Point Encryption (MPPE) [4] is a means of representing PPP packets in an encrypted form. MPPE uses the RSA RC4 [5]
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 [6] in the Compression Control Protocol.

This document describes the method used to derive the initial MPPE session keys from MS-CHAP credentials. The algorithm used to change session keys during a session is described in [4].

3. 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 [7].

4. Deriving Session Keys from MS-CHAP Credentials

The following sections detail the methods used to derive initial session keys (both 40- and 128-bit) from MS-CHAP credentials.

Implementation Note

The initial session key in both directions is derived from the credentials of the peer that initiated the call and the challenge used (if any) is the challenge from the first authentication. This is true for both unilateral and bilateral authentication, as well as for each link in a multilink bundle. In the multi-chassis multilink case, implementations are responsible for ensuring that the correct keys are generated on all participating machines.

4.1. Generating 40-bit Session Keys

MPPE uses a derivative of the peer’s LAN Manager password as the 40-bit session key used for initializing the RC4 encryption tables.

The first step is to obfuscate the peer’s password using the LmPasswordHash() function (described in [3]). The first 8 octets of the result are used as the basis for the session key generated in the following way:

```c
/*
 * PasswordHash is the basis for the session key
 * SessionKey is a copy of PasswordHash and is the generative session key
```
* 8 is the length (in octets) of the key to be generated.
+ */
Get_Key(PasswordHash, SessionKey, 8)

/*
* The effective length of the key is reduced to 40 bits by
* replacing the first three bytes as follows:
*/
SessionKey[0] = 0xD1;
SessionKey[1] = 0x26;
SessionKey[2] = 0x9E;

4.2. Generating 128-bit Session Keys

MPPE uses a derivative of the peer's Windows NT password as the 128-bit
session key used for initializing encryption tables.

The first step is to obfuscate the peer's password using NtPassword-
Hash() function as described in [3]. The first 16 octets of the result
are then hashed again using the MD4 algorithm. The first 16 octets of
the second hash are used as the basis for the session key generated in
the following way:

/*
* Challenge (as described in [7]) is sent by the PPP authenticator
* during authentication and is 8 octets long.
* NtPasswordHashHash is the basis for the session key.
* On return, InitialSessionKey contains the initial session
* key to be used.
*/
Get_Start_Key(Challenge, NtPasswordHashHash, InitialSessionKey)

/*
* CurrentSessionKey is a copy of InitialSessionKey
* and is the generative session key.
* Length (in octets) of the key to generate is 16.
*
*/
Get_Key(InitialSessionKey, CurrentSessionKey, 16)

4.3. Key Derivation Functions

The following procedures are used to derive the session key.

/*
/* Pads used in key derivation */

SHApad1[40] = 
{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

SHApad2[40] = 
{0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2, 0xF2};

/* SHAInit(), SHAUpdate() and SHAFinal() functions are an implementation of Secure Hash Algorithm (SHA-1) [8]. These are available in public domain or can be licensed from RSA Data Security, Inc. */

* 1) InitialSessionKey is 8 octets long for 40 bit session keys, 16 octets long for 128 bit session keys.
* 2) CurrentSessionKey is same as InitialSessionKey when this routine is called for the first time for the session.*

Get_Key(
IN InitialSessionKey,
IN/OUT CurrentSessionKey
IN LengthOfDesiredKey )
{
  SHAInit(Context)
  SHAUpdate(Context, InitialSessionKey, LengthOfDesiredKey)
  SHAUpdate(Context, SHAPad1, 40)
  SHAUpdate(Context, CurrentSessionKey, LengthOfDesiredKey)
  SHAUpdate(Context, SHAPad2, 40)
  SHAFinal(Context, Digest)
  memcpy(CurrentSessionKey, Digest, LengthOfDesiredKey)
}

Get_Start_Key(
IN Challenge,
IN NtPasswordHashHash,
OUT InitialSessionKey)
{
  SHAInit(Context)
  SHAUpdate(Context, NtPasswordHashHash, 16)
SHAUpdate(Context, NtPasswordHashHash, 16)
SHAUpdate(Context, Challenge, 8)
SHAFinal(Context, Digest)
memcpy(InitialSessionKey, Digest, 16)

5. Security Considerations

Because of the way in which 40-bit keys are derived, the initial 40-bit session key will be identical in all sessions established under the same peer credentials. For this reason, and because RC4 with a 40-bit key length is believed to be a relatively weak cipher, peers SHOULD NOT use 40-bit keys derived from the LAN Manager password hash (as described above) if it can be avoided.

6. References


[5] 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


7. 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 (rfriend@hifn.com), Joe Davies (josephd@microsoft.com), Jody Terrill (jodyt@extendsys.com), Archie Cobbs (archie@whistle.com), Mark Deuser (deuser@us.ibm.com), and Jeff Haag (jeff_haag@3com.com) for useful feedback.

8. Chair’s Address

The PPP Extensions Working Group can be contacted via the current chair:

    Karl Fox
    Ascend Communications
    3518 Riverside Drive
    Suite 101
    Columbus, OH 43221

    Phone: +1 614 326 6841
    Email: karl@ascend.com

9. Author’s Address

Questions about this memo can also be directed to:

    Glen Zorn
    Microsoft Corporation
    One Microsoft Way
    Redmond, Washington 98052

    Phone: +1 425 703 1559
    FAX: +1 425 936 7329
    EMail: glennz@microsoft.com

10. Expiration Date

This memo is filed as <draft-ietf-pppext-mschapv1-keys-00.txt> and expires on March 24, 1999.

Appendix A - Sample Key Derivations
The following sections illustrate both 40- and 128-bit key derivations. All intermediate values are in hexadecimal.

**Appendix A.1 - Sample 40-bit Key Derivation**

**Initial Values**
Password = "clientPass"

**Step 1:** LmPasswordHash(Password, PasswordHash)
PasswordHash = 76 a1 52 93 60 96 d7 83 0e 23 90 22 74 04 af d2

**Step 2:** Copy PasswordHash to SessionKey
SessionKey = 76 a1 52 93 60 96 d7 83 0e 23 90 22 74 04 af d2

**Step 3:** GetKey(PasswordHash, SessionKey, 8)
SessionKey = d8 08 01 53 8c ec 4a 08

**Step 4:** Reduce the effective key length to 40 bits
SessionKey = d1 26 9e 53 8c ec 4a 08

**Appendix A.2 - Sample 128-bit Key Derivation**

**Initial Values**
Password = "clientPass"
Challenge = 10 2d b5 df 08 5d 30 41

**Step 1:** NtPasswordHash(Password, PasswordHash)
PasswordHash = 44 eb ba 8d 53 12 b8 d6 11 47 44 11 f5 69 89 ae

**Step 2:** GetStartKey(Challenge, PasswordHashHash, InitialSessionKey)
InitialSessionKey = a8 94 78 50 cf c0 ac ca d1 78 9f b6 2d dc dd b0

**Step 3:** Copy InitialSessionKey to CurrentSessionKey
CurrentSessionKey = a8 94 78 50 cf c0 ac ca d1 78 9f b6 2d dc dd b0

**Step 4:** GetKey(InitialSessionKey, CurrentSessionKey, 16)
CurrentSessionKey = 59 d1 59 bc 09 f7 6f 1d a2 a8 6a 28 ff ec 0b 1e