Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2)

Bruce Schneier
Counterpane Systems
`schneier@counterpane.com`

Mudge
L0pht Heavy Industries
`mudge@l0pht.com`

David Wagner
UC Berkeley
`daw@cs.berkeley.edu`

September 12, 1999

Abstract

The Point-to-Point Tunneling Protocol (PPTP) is used to secure PPP connections over TCP/IP link. In response to [SM98], Microsoft released extensions to the PPTP authentication mechanism (MS-CHAP), called MS-CHAPv2. We present an overview of the changes in the authentication and encryption-key generation portions of MS-CHAPv2, and assess the improvements and remaining weaknesses in Microsoft's PPTP implementation.

1 Introduction

The Point-to-Point Tunneling Protocol (PPTP) [HP+97] is a protocol that allows Point-to-Point Protocol (PPP) connections [Sim94] to be tunneled through an IP network, creating a Virtual Private Network (VPN). Microsoft has implemented its own algorithms and protocols to support PPTP. This implementation of PPTP, called Microsoft PPTP, is used extensively in commercial VPN products precisely because it is already a part of the Microsoft Windows 95, 98, and NT operating systems.

The authentication protocol in Microsoft PPTP is the Microsoft Challenge / Reply Handshake Protocol (MS-CHAP) [ZC98]; the encryption protocol is Microsoft Point to Point Encryption (MPPE) [PZ98]. After Microsoft's PPTP was cryptanalyzed [SM98] and significant weaknesses were publicized, Microsoft upgraded their protocols [Zor98a, Zor98b, Zor99]. The new version is called MS-CHAP version 2 (MS-CHAPv2); the older version has been renamed as MS-CHAP version 1 (MS-CHAPv1). MS-CHAPv2 is available as an upgrade for Microsoft Windows 95, Windows 98, and Windows NT 4.0 (DUN 1.3) [Mic98a, Mic98b]. Even though this upgrade is available, we believe that most implementation of PPTP use MS-CHAPv1.

This paper examines MS-CHAPv2 and discusses how well it addresses the security weaknesses outlined in [SM98].

The most significant changes from MS-CHAPv1 to MS-CHAPv2 are:

- The weaker LAN Manager hash is no longer sent along with the stronger Windows NT hash. This is to prevent automatic password crackers like L0phtcrack [L99] from first breaking the weaker LAN Manager hash and then using that information to break the stronger NT hash [L97].
- An authentication scheme for the server has been introduced. This is to prevent malicious servers from masquerading as legitimate servers.
- The change password packets from MS-CHAPv1 have been replaced by a single change password packet in MS-CHAPv2. This is to prevent the active attack of spoofing MS-CHAP failure packets.
- MPPE uses unique keys in each direction. This is to prevent the trivial cryptanalytic attack of XORing the text stream in each direction to remove the effects of the encryption [SM98].

These changes do correct the major security weaknesses of the original protocol: the inclusion of the LAN Manager hash function and the use of the same OFB encryption key multiple times. However, many security problems are still unaddressed: e.g., how the client protects itself, the fact that the encryption key has the same entropy as the user's password, and the fact that enough data is passed on the wire to allow attackers to mount crypt-and-compare attacks.

This being said, Microsoft obviously took this opportunity to not only fix some of the major cryptographic weaknesses in their implementation of PPTP, but also to improve the quality of their code. The new version is much more robust against denial-of-service style attacks and no longer leaks information regarding the number of active VPN sessions.

2 MS-CHAP, Versions 1 and 2

The MS-CHAPv1 challenge/response mechanism was described in [SM98]. It consists of the following steps:

1. Client requests a login challenge from the Server.
2. The Server sends back an 8-byte random challenge.
3. The Client uses the LAN Manager hash of its password to derive three DES keys. Each of these keys is used to encrypt the challenge. All three encrypted blocks are concatenated into a 24-byte reply. The Client creates a second 24-byte reply using the Windows NT hash and the same procedure.

4. The server uses the hashes of the Client's password, stored in a database, to decrypt the replies. If the decrypted blocks match the challenge, the authentication completes and sends a "success" packet back to the client.

This exchange has been modified in MS-CHAPv2. The following is the revised protocol:

1. Client requests a login challenge from the Server.

2. The Server sends back a 16-byte random challenge.

3a. The Client generates a random 16-byte number, called the "Peer Authenticator Challenge."

3b. The Client generates an 8-byte challenge by hashing the 16-byte challenge received in step (2), the 16-byte Peer Authenticator Challenge generated in step (3a), and the Client's username. (See Section 3 for details.)

3c. The Client creates a 24-byte reply, using the Windows NT hash function and the 8-byte challenge generated in step (3b). This process is identical to MS-CHAPv1.

3d. The Client sends the Server the results of steps (3a) and (3c).

4a. The Server uses the hashes of the Client's password, stored in a database, to decrypt the replies. If the decrypted blocks match the challenge, the Client is authenticated.

4b. The Server uses the 16-byte Peer Authenticator Challenge from the client, as well as the Client's hashed password, to create a 20-byte "Authenticator Response." (See Section 5 for details.)

5. The Client also computes the Authenticator Response. If the computed response matches the received response, the Server is authenticated.

A general description of the changes between MS-CHAPv1 and MS-CHAPv2 is given in Figure 1. This protocol works, and eliminates the most serious weaknesses that plagued MS-CHAPv1. In MS-CHAPv1, two parallel hash values were sent from the Client to the Server: the LAN Manager hash and the Windows NT hash. These were two different hashes of the same User password. The LAN Manager hash is a much weaker hash function, and password-cracker programs such as L0phtcrack were able to break the LAN Manager hash and then use that information to break the Windows NT hash [L97]. By eliminating the LAN Manager hash in MS-CHAPv2, Microsoft has made this divide-and-conquer attack impossible. Still, the security of this protocol is based on the password used, and L0phtcrack can still break weak passwords using a dictionary attack [L99].

As we will discuss later, multiple layers of hashing are used in the different steps of MS-CHAPv2. While this hashing serves to obscure some of the values,

MS-CHAP Version 1	MS-CHAP Version 2
Negotiates CHAP with an algorithm value of `0x80`.	Negotiates CHAP with an algorithm value of `0x81`.
Server sends an 8-byte challenge value.	Server sends a 16-byte value to be used by the client in creating an 8-byte challenge value.
Client sends 24-byte LANMAN and 24-byte NT response to 8-byte challenge.	Client sends 16-byte peer challenge that was used in creating the hidden 8-byte challenge, and the 24-byte NT response.
Server sends a response stating SUCCESS or FAILURE.	Server sends a response stating SUCCESS or FAILURE and piggybacks an Authenticator Response to the 16-byte peer challenge.
Client decides to continue or end based upon the SUCCESS or FAILURE response above.	Client decides to continue or end based upon the SUCCESS or FAILURE response above. In addition, the Client checks the validity of the Authenticator Response and disconnects if it is not the expected value.

Figure 1: Some basic differences between MSCHAPv1 and MSCHAPv2 authentication

it is unclear what the cryptographic significance of them are. All they seem to do is to slow down the execution of the protocol.

We also have concerns over the amount of control the client has in the influence of the ultimate 8-byte challenge that is used, though we have not yet been able to come up with a viable attack to exploit this. Certainly it opens the possibility of subliminal channels, which can be exploited in other contexts.

3 MS-CHAPv2: Deriving the 8-byte Challenge for the 24-byte Response

In MS-CHAPv1, the Server sends the Client an 8-byte random challenge. This challenge is used, together with the Client's password and a hash function, to create a pair of 24-byte responses.

In MS-CHAPv2, the Server sends the Client a 16-byte challenge. This challenge is not used by the Client directly; the Client derives an 8-byte value from this 16-byte challenge. The derivation process is as follows:

1. The Client creates a 16-byte random number, called the Peer Authenticator Challenge.

2. The Client concatenates the Peer Authenticator Challenge with the 16-byte challenge received from the server and the Client's username.

3. The client hashes the result with SHA-1 [NIST93].

4. The first eight bytes of the hash become the 8-byte challenge.

It is these 8 bytes that the Client will use to encrypt the 16-byte local password hash (using the Windows NT hash function) to obtain the 24-byte response, which the Client will send to the server. This method is identical to MS-CHAPv1, and has been described in [SM98].

3.1 Analysis

It is unclear to us why this protocol is so complicated. At first glance, it seems reasonable that the Client not use the challenge from the Server directly, since it is known to an eavesdropper. But instead of deriving a new challenge from some secret information—the password hash, for example—the Client uses a unique random number that is sent to the Server later in the protocol. There is no reason why the Client cannot use the Server's challenge directly and not use the Peer Authenticator Challenge at all.

4 MS-CHAPv2: Deriving the 24-byte Response

Both MS-CHAPv1 and MS-CHAPv2 use the same procedure to derive a 24-byte response from the 8-byte challenge and the 16-byte NT password hash:

1. The 16-byte NT hash is padded to 21 bytes by appending five zero bytes.

2. Let X, Y, Z be the three consecutive 7-byte blocks of this 21-byte value, and let C be the 8-byte challenge. The 24-byte response R is calculated as $R = \langle \mathrm{DES}_X(C), \mathrm{DES}_Y(C), \mathrm{DES}_Z(C) \rangle$.

4.1 Analysis

This complicated procedure creates a serious weakness in the MS-CHAP protocols: it allows the attacker to speed up dictionary keysearch by a factor of 2^{16}, which is a pretty devastating effect given the relatively low entropy of most user passwords.

Suppose that we eavesdrop on a MS-CHAP connection. The response R is exposed in the clear, and the challenge C may be derived easily from public information. We will attempt to recover the password, using the knowledge that many passwords are closely derived from dictionary words or otherwise readily guessable.

Note first that the value of Z can be easily recovered: since there are only 2^{16} possibilities for Z, and we have a known plaintext-ciphertext pair $C, \mathrm{DES}_Z(C)$

for Z, we may try each of the possibilities for Z in turn with a simple trial encryption[1]. This discloses the last two bytes of the NT hash of the password.

We will use this observation to speed up dictionary search. In a one-time precomputation, we hash each of our guesses at the password (perhaps as minor variations on a list of words in a dictionary). We sort the results by the last two bytes of their hash and burn this on a CD-ROM (or a small hard drive). Then, when we see a MS-CHAP exchange, we may recover the last two bytes of the NT hash (using the method outlined above) and examine all the corresponding entries on the CD-ROM. This gives us a list of plausible passwords which have the right value for the last two bytes of their NT hash; then we can try each of those possibilities by brute force.

Suppose a naive dictionary attack would search N passwords. In our attack, we try only the passwords which have the right value for the last two bytes of their NT hash, so we expect to try only about $N/2^{16}$ passwords. This implies that the optimized attack runs about 2^{16} times faster than a standard dictionary attack, if we can afford the space to store a precomputed list of possible passwords.

This attack is applicable to both MS-CHAPv1 and MS-CHAPv2. However, the weakness is much more important for MS-CHAPv2, because for MS-CHAPv1 it is easier to attack the LanManager hash than to attack the NT hash.

This is a serious weakness which could have been easily avoided merely by using a standard cryptographic hashing primitive. For instance, merely generating the response as $R = \text{SHA-1}(\text{NT hash}, C)$ would be enough to prevent this attack.

Note also that the MS-CHAP response generation algorithm is also a weak link, even when passwords contain adequate entropy. It is clear that the NT hash can be recovered with just two DES exhaustive keysearches (about 2^{56} trial DES decryptions on average), or in just 9 days using a single EFF DES Cracker machine [Gil98]. Once the NT hash is recovered, all encrypted sessions can be read and the authentication scheme can be cracked with no effort. This shows that, even when using 128-bit RC4 keys, the MS-CHAP protocol provides at most the equivalent of 57-bit security[2]. This weakness could also have been avoided by the simple change suggested above, $R = \text{SHA-1}(\text{NT hash}, C)$.

It is not clear to us why the MS-CHAPv2 designers chose such a complicated and insecure algorithm for generating 24-byte responses, when a simpler and more secure alternative was available.

---

[1]This has been independently observed by B. Rosenburg.

[2]This has been independently observed by P. Holzer.

5 MS-CHAPv2: Deriving the 20-byte Authenticator Response

In MS-CHAPv2, the Server sends the Client a 20-byte Authenticator Response. The Client calculates the same value, and then compares it with the value received from the Server in order to complete the mutual authentication process. This value is created as follows:

1. The Server (or the Client) hashes the 16-byte NT password hash with [Riv91] to get password-hash-hash. (The Server stores the clients password hashed with MD4; this is the NT password hash value.)

2. The Server then concatenates the password-hash-hash, the 24-byte NT response, and the literal string "Magic server to client constant", and then hashes the result with SHA.

3. The Server concatenates the 20-byte SHA output from step (2), the initial 8-byte generated challenge (see Section 3) and the literal string "Pad to make it do more than one iteration", and then hashes the result with SHA.

The resulting 20 bytes are the mutual authenticator response.

5.1 Analysis

Again, this process is much more complicated than required. There is no reason to use SHA twice; a single hashing has the same security properties.

6 Analysis of MS-CHAPv2

We do not know why Microsoft chose such a complicated protocol, since this is not stronger than the following:

1. The Server sends the Client an 8-byte challenge.

2. The Client encrypts the 16-byte local password hash with an 8-byte challenge and sends the Server the 24-byte response, an 8-byte challenge of its own, and the username.

3. The Server sends a pass/fail packet with a 24-byte response to the Client's challenge, which is the user's password-hash-hash encrypted with the Client's 8-byte challenge.

The downside to the MS-CHAPv2 protocol is that an eavesdropper can obtain two copies of the same plaintext, encrypted with two different keys. However, in the current model, watching the network for any length of time will still give you multiple copies of a user challenge/response as the user logs in and out, which will be encrypted with different keys.

As it stands, a passive listener is still able to get the 8-byte challenge and the 24-byte response from the information sent. The popular hacker tool L0phtcrack [L97], which breaks Windows NT passwords, works with this data as input. This task was much easier with MS-CHAPv1, since the weaker LAN Manager hash was sent alongside the stronger Windows NT hash; L0phtcrack first broke the former and then used that information to break the latter [L99]. L0phtcrack can still break most common passwords from the Windows NT hash alone [L97]. And this still does not solve the problem of using the user's hash for MPPE keying, PPTP authentication, etc. without negotiating, at least, machine public key/private key methods of exchanging such an important key.

6.1 Version Rollback Attacks

Since Microsoft has attempted to retain some backwards compatibility with MS-CHAPv1, it is possible for an attacker to mount a "version rollback attack" against MS-CHAP. In this attack, the attacker convinces both the Client and the Server not to negotiate the more secure MS-CHAPv2 protocol, but to use the less secure MS-CHAPv1 protocol.

In its documentation, Microsoft claims that the operating systems will try to negotiate MS-CHAPv2 first, and only drop back to MS-CHAPv1 if the first negotiation fails [Mic99]. Additionally, it is possible to set the Server to require MS-CHAPv2. We find this scenario implausible for two reasons. One, the software switches to turn off backwards compatibility are registry settings, and can be difficult to find. And two, since older versions of Windows cannot support MS-CHAPv2, backwards compatibility must be turned on if there are any legacy users on the network. We conclude that version rollback attacks are a significant threat.

7 Changes to MPPE

The original encryption mechanism in Microsoft's Point to Point Encryption protocol (MPPE) used the same encryption keys in each direction (Client to Server, and Server to Client). Since the bulk data encryption routine is the RC4 stream cipher [Sch96], this created a cryptographic attack by XORing the two streams against each other and performing standard cryptanalysis against the result.

In the more recent version, the MPPE keys are derived from MS-CHAPv2 credentials and a unique key is used in each direction. The keys for each direction are still derived from the same value (the Client's NT password hash), but differently depending on the direction.

7.1 Deriving MPPE Keys from MS-CHAPv2 Credentials

MPPE keys can be either 40 bits or 128 bits, and they can be derived from either MS-CHAPv1 credentials or MS-CHAPv2 credentials. The original derivation protocol (from MS-CHAPv1) was described in [SM98]. Briefly, the password hash is hashed again using SHA, and then truncated. For a 40-bit key, the SHA hash is truncated to 64 bits, and then the high-order 24 bits are set to `0xD1269E`. For a 128-bit key, the SHA hash is truncated to 128 bits. This key is used to encrypt traffic from the Client to the Server and traffic from the Server to the Client, opening a major security vulnerability. This has been corrected in MS-CHAPv2.

Deriving MPPE keys from MS-CHAPv2 credentials works as follows:

1. Hash the 16-byte NT password hash, the 24-byte response from the MS-CHAPv2 exchange, and a 27-byte constant (the string "This is the MPPE Master Key") with SHA. Truncate to get a 16-byte master-master key.

2. Using a deterministic process, convert the master-master key to a pair of session keys.

For 40-bit session keys, this is done as follows:

1. Hash the master-master key, 40 bytes of `0x00`, an 84-byte constant and 40 bytes of `0xF2` with SHA. Truncate to get an 8-byte output.

2. Set the high-order 24 bits of `0xD1269E`, resulting in a 40-bit key.

The magic constants are different, depending on whether the key is used to encrypt traffic from the Client to the Server, or from the Server to the Client.

For 128-bit session keys, the process is as follows:

1. Hash the master-master key, 40 bytes of `0x00`, an 84-byte constant (magic constant 2 or 3), and 40 bytes of `0xF2` with SHA. Truncate to get a 16-byte output.

7.2 Analysis

This modification means that unique keys are used in each direction, but does not solve the serious problem of weak keys. The keys are still a function of the password, and hence contain no more entropy than the password. Even though the RC4 algorithm may theoretically have 128-bits of entropy, the actual passwords used for key generation have much less. This having been said, using different keys in each direction is still a major improvement in the protocol.

7.3 Trapdoors in the magic constants?

We are very concerned with the magic constants embedded in the key derivation algorithm for export-weakened keys.

The protocol weakens RC4 keys to 40 bits by fixing the high bits of the 64-bit RC4 key to `0xD1269E`. But this seems dangerous. It is known that, if an adversary is allowed to choose the high bits of the RC4 key, the adversary can force you into a weak key class for RC4 [Roo95, Wag95]. Therefore, if the MS-CHAP designers—or the NSA export-reviewer folks—wanted to embed a trapdoor in the protocol, they could exploit the presence of magic constants to weaken RC4.

We do not know whether keys prefixed with `0xD1269E` are unusually weak, but in our preliminary statistical tests we have found some suspicious properties of such keys that leaves us with some cause for concern. To give two examples:

- Empirical measurements show that the first few bytes of output are biased, for keys which start with `0xD1269E`. The first and second keystream bytes take on the values `0x09` and `0x00` with probabilities 0.0054 and 0.0060, respectively. This is noticeably higher than the $1/256 = 0.0039$ probability you'd expect from good cipher.
- The key schedule mixes some entries in the state table poorly, for this class of keys. For instance, $S[1] =$ `0xF8` holds with probability $0.38 \approx 1/e$, and $S[2] =$ `0x98` holds with a similar probability.

These statistical properties are worrisome.

Because no information is given on how the value `0xD1269E` was chosen, one has to worry that it could well be a "trapdoor choice" which forces all 40-bit keys into some weak key class for RC4. We invite the MS-CHAP designers to openly disclose how all magic constants were chosen and to provide concrete assurances that those magic values don't create any hidden trapdoors. In the meantime, we leave it as an open question to ascertain whether RC4 is secure when used with the fixed key-prefix `0xD1269E`.

8 Attack on export-weakened key derivation

In this section we present a very serious attack on the way that exportable 40-bit session keys are generated. This weakness is also present in MS-CHAPv1 as well as MS-CHAPv2, but it has not been discovered until now.

The end result is that the so-called "40-bit keys" really only have an effective strength of about 26 bits. As a result, the export-weakened protocol can be cracked in near-realtime with only a single computer[3].

We recall that the key derivation process appends 40 secret bits (generated in some way which is irrelevant to our attack) to the fixed value `0xD1269E`. The resulting 64-bit session key is to RC4-encrypt the transmitted data. The problem is that this process introduces no per-session salt (compare to, e.g., SSL), and thus can be broken with a time-space tradeoff attack.

For the remainder of this section, we assume that we can obtain a short segment of known plaintext (40 bits should suffice) at some predictable location. The known plaintext need not even occur at consecutive bit locations; the only requirement is that the bit positions be predictable in advance. This seems to be

[3] Today's computers seem to be able to try 2^{16}–2^{17} keys/second, which suggests that each key can be cracked in something like a quarter of an hour. (In lieu of an implementation, these estimates will necessarily be very rough.) With a small cluster of computers, the cracking performance can be greatly increased.

a very plausible assumption, when one considers the quantity of known headers and other predictable data that is encrypted. Let us assume for simplicity of description that this known plaintext occurs at the start of the keystream.

We will attack this protocol with a time-space tradeoff. The cost of a lengthy precomputation is amortized over many sessions so that the incremental cost of breaking each additional session key is reduced to a very low value.

A naive attacker might consider building a lookup table with 2^{40} entries, listing for each possible 40-bit key the value of the first 40 bits of keystream that results. This requires a 2^{40} precomputation, but then each subsequent session key can be broken extremely quickly (with just a single table lookup). However, in practice this attack is probably not very practical because it requires 2^{40} space.

A time-space tradeoff allows us to reduce the space requirements of the naive attack by trading off memory for additional computation. Consider Hellman's time-space tradeoff [Hel80]. For a n-bit key, Hellman's tradeoff requires a 2^n precomputation and $2^{2n/3}$ space, and then every subsequent session key can be broken with just $2^{2n/3}$ work. (Other tradeoffs are also possible.)

For MS-CHAP's 40-bit keys, $n = 40$, and $2n/3 \approx 26$, so you get an attack that breaks each session key with approximately 2^{26} work. The attack requires a 2^{40} precomputation and 2^{26} space, but these requirements are easily met.

This means that the export-weakened versions of MS-CHAP offer an effective keylength of only about 26 bits or so, which is much less than the claimed 40 bits of strength. This is a deadly weakness.

9 Conclusions

Microsoft has improved PPTP to correct the major security weaknesses described in [SM98]. However, the fundamental weakness of the authentication and encryption protocol is that it is only as secure as the password chosen by the user. As computers get faster and distributed attacks against password files become more feasible, the list of bad passwords—dictionary words, words with random capitalization, words with the addition of numbers, words with numbers replacing letters, reversed words, acronyms, words with the addition of punctuation—becomes larger. Since authentication and key-exchange protocols which do not allow passive dictionary attacks against the user's password are possible—Encrypted Key Exchange [BM92, BM94] and its variants [Jab96, Jab97, Wu98], IPSec—it seems imprudent for Microsoft to continue to rely on the security of passwords. Our hope is that PPTP continues to see a decline in use as IPSec becomes more prevalent.

References

[BM92] S.M. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks," *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 1992, pp. 72–84.

[BM94] S.M. Bellovin and M. Merritt, "Augmented Encrypted Key Exchange: A Password-Based Protocol Secure Against Dictionary Attacks and Password File Compromise," AT&T Bell Laboratories, 1994.

[Gil98] J. Gilmore, Ed., *Cracking DES*. The Electronic Frontier Foundation, San Francisco, CA, O'Reilly and Associates, 1998.

[HP+97] K. Hamzeh, G.S. Pall, W. Verthein, J. Taarud, and W.A. Little, "Point-to-Point Tunneling Protocol," Internet Draft, IETF, Jul 1997. `http://www.ietf.org/internet-drafts/draft-ietf-pppext-pptp-10.txt`.

[Hel80] M.E. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Transactions on Information Theory*, vol.IT-26, no.4, July 1980, p.401–406.

[Jab96] D. Jablon, "Strong Password-Only Authenticated Key Exchange," *ACM Computer Communications Review*, Oct 96, pp. 5–26.

[Jab97] D. Jablon, "Extended Password Key Exchange Protocols Immune to Dictionary Attacks," *Proceedings of the Sixth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, IEEE Computer Society, 1997, pp. 248–255.

[L97] L0pht Heavy Industries, Inc., "A L0phtCrack Technical Rant," Jul 1997. `http://www.l0pht.com/l0phtcrack/rant.html`.

[L99] L0pht Heavy Industries, Inc, L0phtcrack, 1999, `http://www.l0pht.com/l0phtcrack/`.

[Mic96a] Microsoft Corporation, Advanced Windows NT Concepts, New Riders Publishing, 1996. Relevant chapter at `http://www.microsoft.com/communications/nrpptp.htm`.

[Mic96b] Microsoft Corporation, "Point-to-Point Tunneling Protocol (PPTP) Frequently Asked Questions," Jul 1996.

[Mic98a] Microsoft Corporation, "Frequently Asked Questions about Microsoft VPN Security," Dec 1998, `http://www.microsoft.com/NTServer/commserv/deployment/moreinfo/VPNSec_FAQ.asp`

[Mic98b] Microsoft Corporation, "Microsoft Windows 95 Dial-Up Networking 1.3 Upgrade Release Notes," 1998, `http://support.microsoft.com/support/kb/articles/q154/0/91.asp`

[Mic99] Microsoft, Corporation, "Windows 98 Dial-Up Networking Security Upgrade Release Notes," Feb 1999, `http://support.microsoft.com/support/kb/articles/Q189/7/71.asp`.

[NIST93] National Institute of Standards and Technology, "Secure Hash Standard," U.S. Department of Commerce, May 1993.

[PZ98] G.S. Pall and G. Zorn, "Microsoft Point-to-Point Encryption (MPPE) Protocol," Network Working Group, Internet Draft, IETF, Mar 1998. `http://www.ietf.org/internet-drafts/draft-ietf-pppext-mppe-03.txt`.

[Riv91] R. Rivest, "The MD4 Message Digest Algorithm," Advances in Cryptology—CRYPTO'90 Proceedings, Springer-Verlag, 1991, pp. 303311.

[Roo95] A. Roos, "Weak Keys in RC4," sci.crypt post, 22 Sep 1995.

[Sim94] W. Simpson, "The Point-to-Point Protocol (PPP)," Network Working Group, STD 51, RFC 1661, Jul 1994. `ftp://ftp.isi.edu/in-notes/rfc1661.txt`.

[Sch96] B. Schneier, Applied Cryptography, 2nd Edition, John Wiley & Sons, 1996.

[SM98] B. Schneier and Mudge, "Cryptanalysis of Microsofts Point-to-Point Tunneling Protocol (PPTP)," *Proceedings of the 5th ACM Conference on Communications and Computer Security*, ACM Press, pp. 132–141. `http://www.counterpane.com/pptp.html`.

[Wag95] D. Wagner, "Re: Weak Keys in RC4," sci.crypt post, 25 Sep 1995. `http://www.cs.berkeley.edu/~daw/my-posts/my-rc4-weak-keys`.

[Wu98] T. Wu, "The Secure Remote Password Protocol," *Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium*, Mar 1998, pp. 97–111.

[ZC98] G. Zorn and S. Cobb, "Microsoft PPP CHAP Extensions," Network Working Group Internet Draft, Mar 1998. `http://www.ietf.org/internet-drafts/draft-ietf-pppext-mschap-00.txt`.

[Zor98a] G. Zorn, "Deriving MPPE Keys from MS-CHAP V1 Credentials," Network Working Group Internet Draft, Sep 1998. `http://www.ietf.org/internet-drafts/draft-ietf-pppext-mschapv1-keys-00.txt`.

[Zor98b] G. Zorn, "Deriving MPPE Keys from MS-CHAP V2 Credentials," Network Working Group Internet Draft, Nov 1998. `http://www.ietf.org/internet-drafts/draft-ietf-pppext-mschapv2-keys-02.txt`.

[Zor99] G. Zorn, "Microsoft PPP CHAP Extensions, Version 2," Network Working Group Internet Draft, Apr 1999. `http://www.ietf.org/internet-drafts/draft-ietf-pppext-mschap-v2-03.txt`.