A Security Analysis of UBC Wireless Network

Wing-Keong Woo, Qiang Wei, Joyce Hsien-Yin Chiang, and Johnson Ming-Che Tsai

***Abstract*— The security policies of the UBC wireless network are concerned with origin integrity, i.e. authentication, and availability. To enforce origin integrity, users of the UBC wireless network are required to login via one of the three types of authentication mechanisms, known as Quick-Connect, VPN-PPTP and VPN-IPsec. If Quick-Connect is used, we analyzed that an adversary is able to use the authorized user's login to access the network without his knowledge. If VPN-PPTP is used, we analyzed that an adversary may be able to recover the authorized user's account name and password, and hence will be able to access the network. We analyzed that the group password used in the VPN-IPsec may be recovered, which may then lead to a man-in-the-middle and further attacks. To enforce availability, monitoring systems are setup to log users' consumption of bandwidth. However, the correctness of the log is dependent on the correctness of the authentication mechanisms, which are vulnerable. Hence the log may be incorrect and may lead to actions taken against innocent users for over-consumption of bandwidth. We conclude that the security policies may be breached due to the vulnerabilities of the authentication mechanisms, and follow up with discussions to rectify them.**

***Index Terms*—Computer network security, wireless LAN.**

I. Introduction

UBC has setup a campus-wide wireless network primarily intended for teaching, research and administrative purposes [21]. It is free for faculty, students and staff of UBC. It is also free for guests who are sponsored by a faculty or staff of UBC. However, others are not allowed to access the wireless network.

As seen from the UBC Wireless website [34], the security policy is mainly concerned with origin integrity; specifically, only authorized users are allowed to use the wireless network to access the campus resources and Internet. In addition, availability is a concern and authorized users are not allowed to hog the bandwidth of the network, e.g. by running P2P software for heavy and continual file sharing. Although confidentiality is important, it is not a concern of the UBC wireless network, and users are advised to protect themselves if need be.

The origin integrity requirement of the security policy is enforced by the implementation of authentication mechanisms, where users are required to login to the wireless network with the use of their Campus-Wide Login (CWL) accounts and passwords. To cater for novice users to security-conscious users, three types of authentication mechanisms are supported.

The first type of authentication mechanism is known as Quick-Connect, for the fact that users only need to start their web browsers, and will be automatically redirected to the secure UBC Wireless login page when trying to access any website for the first time. Once correct CWL account name and password are entered, a user will be authenticated and be allowed to continue accessing the wireless network and the Internet.

The second type of authentication mechanism is through the use of one of the Virtual Private Network (VPN) technologies, known as Point-to-Point Tunneling Protocol (PPTP) [13], hence we called it VPN-PPTP. Being developed and supported by Microsoft, the VPN-PPTP client software is widely available in Windows. So, to use this authentication mechanism, most users only need to perform a one-time configuration of the VPN-PPTP client software as shown in the UBC Wireless website [34]. Then, to access the UBC wireless network, users only need to launch the VPN-PPTP client software, and enter their CWL account names and passwords to login.

The third type of authentication mechanism is through the use of another VPN technology, known as IPsec [17], hence we called it VPN-IPsec. In this case, users are required to download a VPN-IPsec client software, known as Contivity, from the UBC Wireless website [34]. Upon installation, users will be able to launch the Contivity client software, and enter their CWL account names and passwords to login.

To enforce the availability requirement of the policy, automated monitoring systems are setup to log bandwidth consumption. If a user is detected to consume the bandwidth excessively, he may receive a notification letter which he will need to address. Failing which he may be barred from accessing the network, and/or face disciplinary action resulting in a possible fine.

From the UBC's perspective, if a breach of the security policies is possible, then UBC may be "loosing" the valuable wireless bandwidth to unauthorized users without knowing it. From the users' perspective, if a breach of the security policies is possible, then there will be a possibility of an unauthorized user using an authorized account without his knowledge. In the worst case, if the unauthorized user consumes too much of the bandwidth, the authorized user may be made to face the possible disciplinary action which he did not commit.

Hence, it will be useful to perform a security analysis of the UBC wireless network. Our analysis and experiments show that the authentication mechanisms may be vulnerable, and are elaborated in Sections II, III and IV. Section V concludes with a discussion to rectify the vulnerabilities.

II. Authentication Mechanism 1: Quick-Connect

A. Description of Quick-Connect Authentication

Quick-Connect authentication is based on web technologies protected by SSL3.0 [12]. Wagner et al. [35] have performed an analysis of SSL3.0 and conclude that it is secure.

Basically, three components are involved in the authentication: a client that requires access to the campus resources and Internet, a gateway that control the access, and the UBC Wireless login server that tells the gateway what filtering rules to apply. As shown in Fig. 1, a typical authentication session proceeds as follows:

1. A user launches a web browser and attempts to access a website.
2. As the user is not authenticated yet, his request will be blocked by the gateway. The gateway will then redirect the browser to connect to the UBC Wireless login server using SSL3.0 for authentication.
3. The user's CWL account name and password are protected by SSL3.0 during the authentication.
4. Once the user is authenticated, the gateway will be informed to modify its filtering rules to allow access.
5. Next, the user's web browser is redirected to connect to its original requested website.
6. Since the user is now authenticated, his request is allowed to pass through the gateway, and hence is able to access the campus resources and Internet.

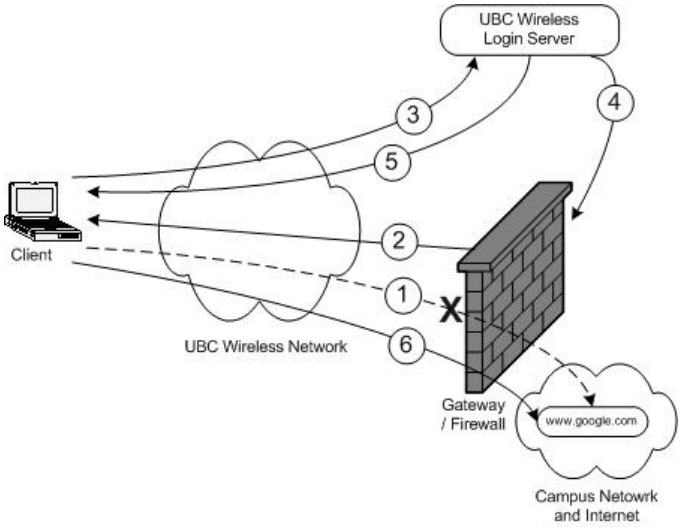

Fig. 1. Quick-Connect authentication process

B. Vulnerability of Quick-Connect Authentication

This authentication mechanism is user-friendly and is likely to be used by many novice users. SSL3.0 is secure, and hence, the user's CWL account name and password are well protected. However, the filtering rules at the gateway are based on MAC and IP addresses, which are not protected. If these addresses can be spoofed, then an adversary will be able to access the network using an existing authorized user's login.

To validate our analysis, we conducted a simple experiment. We setup two Windows laptops, one simulating an authorized user, and another an adversary. The adversary's laptop is installed with Ethereal [9] and SMAC [32] software. Fig. 2 illustrates how an adversary is able to access the Internet without having to authenticate himself:

1. An authorized user login using the Quick-Connect authentication mechanism, and is allowed to access the wireless network and Internet.
2. By running the Ethereal software [9] in promiscuous mode, an adversary is able to capture all packets within range. As MAC and IP addresses are not protected, the adversary is able to determine the authorized user's addresses from the captured packets.
3. The adversary is able to change his IP address to that of the authorized user easily in Windows. By updating the Windows registry manually, or by using tool like SMAC [32], an adversary is also able to change his MAC address to that of the authorized user.
4. After restarting Windows, the adversary's laptop will have the same MAC and IP addresses of the authorized user, and is able to access the Internet as long as the authorized user is logged on.

As expected, our experiment ran smoothly. The adversary is able to access the network, and the authorized user is not aware of that!

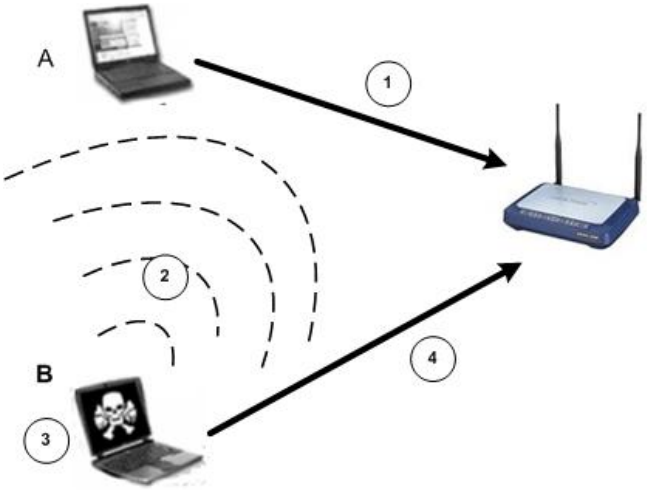

Fig. 2. An adversary performing a MAC and IP spoofing attack to access the network.

III. Authentication Mechanism 2: VPN-PPTP

A. Description of VPN-PPTP Authentication

VPN-PPTP relies on MS-CHAPv2 [37] to perform mutual authentication. The protocol uses SHA-1 [23], MD-4 [28] and DES [24] cryptographic algorithms. Essentially, three messages are exchanged between the VPN-PPTP client software and the UBC wireless VPN server in three steps as follows: (For clarity, data items not required for the understanding of the authentication protocol are omitted.)

1. server -> client: svrChal

2. client -> server: clntChal || clntRes || usrName

3. server -> client: statusCode || svrRes

Step 1 is the CHAP Challenge. The server randomly generates a 16-byte svrChal, and sends it to the client. This serves as a nonce to prevent replay attack.

Step 2 is the CHAP Response. Similar to the server, the client also randomly generates a 16-byte clntChal. The usrName is the CWL account name entered by the user. The client then sends clntChal, clntRes and usrName to the server, where clntRes is derived as follows:

a. Compute chalHash = first 8 bytes (SHA-1 (clntChal || svrChal || usrName).

b. Compute usrPasswdHash = MD4 (usrPasswd in unicode), where usrPasswd is the CWL password entered by the user, which is then converted to unicode representation.

c. Append 5 bytes of 0x00 to the end of usrPasswdHash and split them as follows:

x = first 7 bytes (usrPasswdHash || 00 00 00 00 00)

y = next 7 bytes (usrPasswdHash || 00 00 00 00 00)

z = last 7 bytes (usrPasswdHash || 00 00 00 00 00)

d. Perform DES encryption of chalHash under keys x, y and z separately as follows:

Ex = DESx (chalHash)

Ey = DESy (chalHash)

Ez = DESz (chalHash)

e. Finally, clntRes = Ex || Ey || Ez

Step 3 is the CHAP Status. With clntChal and usrName received from the client, the server uses its own authentic copy of usrPasswdHash to perform the same operations as the client and derive its expected clntRes'. If clntRes' equals clntRes received from the client, the user is considered authenticated since only the authentic user will know the correct password. The statusCode will then be set to success. Otherwise, the statusCode will be set to failure. To authenticate server to the client, the server returns a svrRes derived as follows:

a. Compute usrPasswdHashHash = MD4 (usrPasswdHash)

b. Compute svrHash = SHA-1 (usrPasswdHashHash || clntRes || "Magic server to client signing constant")

c. Compute svrRes = SHA-1 (svrHash || svrChal || "Pad to make it do more than one iteration")

Upon receiving svrRes, the client will use its own copy of usrPasswd entered by the user to derive its expected svrRes'. If svrRes' equals svrRes received from the server, the server is considered authenticated since only the authentic server will know the correct password.

After authentication is successful, MS-MPPE keys are derived [38] to protect subsequent communications between the client and the server. Without knowing the MS-MPPE keys, simple MAC and IP spoofing attack as described in Section III cannot be applied here.

B. Vulnerability of VPN-PPTP Authentication

Schneier et al. [30][31] had analyzed the VPN-PPTP and concluded that it is vulnerable.

To verify the analysis, we perform an experiment as follows:

1. By using Ethereal software [9], we are able to sniff a VPN-PPTP authentication session, and recover svrChal, clntChal, clntRes and usrName.

2. Hence we are able to compute chalHash = first 8 bytes (SHA-1 (clntChal || svrChal || usrName).

3. By splitting clntRes, we obtained Ex, Ey and Ez.

4. Therefore we have the plaintext chalHash, and the corresponding ciphertext Ex, Ey and Ez enciphered under keys x, y and z respectively.

5. Knowing that the last 5 bytes of key z are 0x00, recovering key z is trivial with brute-force key search of 14-bit key.

6. If brute-force key search of 56-bit key is also feasible, which we will discuss shortly, then we will be able to recover key x and y.

7. Then we will have the usrPasswdHash. By performing a dictionary attack, say using L0phtCrack [19], we may be able to recover the user password.

The use of 56-bit key for DES has been controversial since its adoption as an encryption standard. As early as 1977, W. Diffie and M. Hellman [6] estimated that a $20M machine can be constructed to recover the key in a day.

In 1997, Biham [3] measured the performance of a fast DES implementation in software. A 32-bit 133MHz RISC processor was able to perform 2^{18} decryptions per second. So, it is estimated that it will take an average 4000 years to do a brute-force key search of 56-bit DES.

To determine the strength of DES in practice, RSA launched a series of DES cracking competition in late 90s. In Jan 1999, the last DES Challenge III was won in just 22 hours 15 minutes by distributed.net [7], which coordinated the use of idle CPU time of around 100,000 computers connected to the internet world-wide, and a EFF DES cracker machine [8].

In mid 2004, a professor in a Finland university gave an optional assignment [27] for students to do a brute-force key search of 56-bit DES with 8 bits of the key known. By writing a distributed DES software and soliciting help in their website, a group of students was able to break it in about a month using 383 volunteering computers from the Internet. Assuming the same 383 computers were used, which took an average 1 month to perform brute-force key search of 48-bit DES, it would take another 256 months, or about 20 years, to brute-force the full 56-bit DES.

To verify the effort, we wrote a raw version of DES program in C, compiled and ran it under Windows XP with a 2.0GHz Pentium processor. Our implementation was able to perform 2^{21} decryptions in 14 seconds. So, for a single computer, it will take about 7600 years to do a brute-force key search. If we have 383 such computers, it will also take an average 20 years. Our implementation will perform better if time is permitted to optimize our codes.

Hence, even for low-budget adversary, if he is committed and is able to garner help from thousands of computers in the Internet (which is an obtainable goal), it is possible to perform

brute-force key search within a reasonable time.

Coupled with the fact that a user is allowed to have a weak password for CWL account, and not have to change it even for years, the probability of an adversary successfully recovering a password may not be negligible.

IV. Authentication Mechanism 3: VPN-IPsec

A. Description of VPN-IPsec Authentication

Authentication in VPN-IPsec is done in two phases. Phase 1 uses Diffie-Hellman key-exchange algorithm [5] to derive a secret key, which is then used to protect phase 2 authentication. There are a few options available. To be clear and specific, we shall only describe the features implemented in the UBC wireless network.

In Phase 1, the Contivity client software uses a group ID (grpId) and password (grpPasswd), which come preinstalled with the Contivity software to perform mutual authentication with the UBC wireless IPsec server. The protocol uses the Aggressive Mode of the Internet Key Exchange protocol [14] over the ISAKMP [22]. HMAC-MD5 [18][29] and 3DES-CBC [24] cryptographic algorithms are used. Three messages are exchanged between client and server in three steps as follows: (Again, for clarity, data items not necessary for the understanding of the protocol are omitted.)

1. client -> server: CKY-I || SAi_b || g^x || Ni_b || IDii_b
2. server -> client: CKY-R || SAr_b || g^y || Nr_b || IDir_b || Hash_r
3. client -> server: (Hash_i)

In Step 1, the client randomly generates a 8-byte cookie CKY-I meant to prevent denial-of-service attack. It also randomly generates a 20-byte nonce to prevent replay attack, which is packed in a data structure Ni_b, called nonce payload. In addition, it randomly generates a client secret x, and computes g^x according to the Diffie-Hellman key-exchange algorithm [5]. The proposed cryptographic algorithms to be used, e.g. MD5 and 3DES-CBC, are packed into a data structure SAi_b, called security association payload. To prevent grpId from transmitted in clear, it is hashed using SHA-1 [23] accordingly to algorithm specified in [20], and packed into a data structure IDii_b, called identification payload. Finally, these are sent to the server.

In Step 2, the server also randomly generates a 8-byte cookie CKY-R and a 20-byte nonce packed in Nr_b. Similarly, it randomly generates a server secret y and computes g^y. The accepted cryptographic algorithms is packed in SAr_b, which is selected from the list proposed by client in SAi_b. The IP address is the identity of the server and is packed in IDir_b. These, together with Hash_r, are sent to the client. Hash_r is computed as follows:

a. Compute pre-shared-key = SHA1 (grpPasswd, grpId), where grpPasswd and grpId are presumed only known to the Contivity client software and IPsec server, not even the authorized users.

b. Compute SKEYID = HMAC_MD5(pre-shared-key, Ni_b || Nr_b)

c. Compute Hash_r = HMAC_MD5(SKEYID, g^y || g^x || CKY-R || CKY-I || SAr_b || IDir_b)

In Step 3, the client uses its own copy of grpId and grpPasswd in the Contivity client software to compute its pre-shared-key and the expected Hash_r'. If Hash-r' equals the Hash_r received from the server, then the server is considered authenticated since only the authentic server will know the grpId and grpPasswd. Next, to authenticate itself to the server, the client will compute Hash_i = HMAC_MD5(SKEYID, g^x || g^y || CKY-I || CKY-R || SAi_b || IDii_b). The Hash_i is encrypted using 3DES-CBC with key SK as derived below.

Since both client and server have g^x and g^y after step 2, they are able to compute a common secret g^xy with their secret copies of x and y respectively. After that, a common secret key SK to be used for protection subsequent communications is computed as follows:

a. Compute SKEYID_d = HMAC_MD5 (SKEYID, g^xy || CKY-I || CKY-R || 0)

b. Compute SKEYID_a = HMAC_MD5 (SKEYID, SKEYID_d || g^xy || CKY-I || CKY-R || 1)

c. Compute SKEYID_e = HMAC_MD5 (SKEYID, SKEYID_a || g^xy || CKY-I || CKY-R || 2)

d. Compute K1 = HMAC_MD5(SKEYID_e, 0)

e. Compute K2 = HMAC_MD5(SKEYID_e, K1)

f. Compute SK = first 24 bytes (K1 || K2)

The server will also compute its expected Hash_i' with its copy of grpId and grpPasswd. Upon receiving encrypted (Hash_i), the server will first decrypt it to recover Hash_i. If Hash-i' equals Hash_i, then the client is considered authenticated since only the Contivity client will know the grpId and grpPasswd.

In Phase 2, the Contivity client software uses the user CWL account name and password to authenticate to the server, protected by 3DES-CBC encryption with key SK established in the first phase. The Contivity software is likely to implement Xauth [2] authentication. However, we were not able to find any authoritative document describing it. We were also not able to trace it since the communications are all encrypted. Hence, we have to stop our analysis at Phase 1. Nevertheless, we discover a potential vulnerability in Phase 1 as described next.

B. Vulnerability of VPN-IPsec Authentication

Researchers [10][26] who have attempted analyzing IPsec had all commented about its complexity, and cautioned that it may not be fully analyzed for vulnerability. Extensions to support traditional user name/password authentication have also introduced more complexity and the potential for vulnerability.

We perform an experiment as follows:

1. By using Ethereal software [9], we are able to sniff and recover CKY-I, g^x, Ni_b, IDii_b, CKY-R, SAr_b, g^y, Nr_b, IDir_b and Hash_r.

2. The Contivity client software is freely available at the UBC wireless website for all users to download, including the adversary. By looking inside the Contivity client software, the

Group ID can be seen. For UBC Wireless, the grpId is 7uXuCruT8u.

3. We compute SHA-1(grpId), compare it with IDii_b, and verified that Contivity is indeed implementing identity obfuscation according to the expired Internet Draft [20].

4. Since Hash_r is available, we may be able to perform a dictionary attack to recover grpPasswd as follows:

a. Guess a grpPasswd

b. Compute pre-shared-key = SHA1 (grpPasswd, grpId)

c. Compute SKEYID = HMAC_MD5 (pre-shared-key, Ni_b || Nr_b)

d. Compute Hash_r' = HMAC_MD5 (SKEYID, g^y || g^x || CKY-R || CKY-I || SAr_b || IDir_b)

e. If Hash_r' equals Hash_r, we have recovered grpPasswd. Otherwise, repeat from (a).

5. Again by looking at the Contivity client software, we suspect that the grpPasswd is of 10 characters, since there are 10 asterisks shown. This information will greatly assist and make the dictionary attack more feasible.

If the grpPasswd can be recovered, an adversary will be able to launch a man-in-the-middle attack. Although it is computationally infeasible to attack Diffie-Hellman key-exchange algorithm to recover key SK, an adversary may spoof as an authentic UBC Wireless server. Then the adversary may launch further attacks in Phase 2 of the authentication with an attempt to recover the user's CWL account name and password.

V. Discussions

For authentication mechanism 1: Quick-Connect, the vulnerability can be attributed to the failure to follow the principle of complete mediation. As a result, after an authorized user has been authenticated successfully, an adversary is able to perform MAC and IP spoofing attack to access the network.

The wireless system has a built-in security feature called Wired-Equivalent Privacy (WEP) [15]. If enabled, it authenticates a client machine with a pre-shared key. Once authenticated, a common secret key is then derived which is used to protect subsequent communications, thus preventing the simple MAC and IP spoofing attack. Nevertheless, the design of WEP is flawed and various vulnerabilities were found [4][11][33]. There even exists an Airsnort tool [1] that allows an adversary to passively recover the pre-shared key.

There seems no easy solution to prevent MAC and IP spoofing attack when Quick-Connect authentication mechanism is used, except to stop supporting this mechanism totally.

For authentication mechanism 2: VPN-PPTP, the vulnerability exists mainly because the designer of the authentication protocol did not keep up with time to improve it.

Schneier et al. [31] had commented on the unnecessary complicated but vulnerable MS-CHAPv2 protocol, and suggested ways to improve it. However, it seems that Microsoft is not convinced of the vulnerability to improve it.

To reduce the probability for an adversary to recover a user's password, password aging may be considered for implementation. This will ensures that a user will change his password after some time has passed, say 3 months. In addition, proactive password checking [36] may be implemented to ensure that users will not choose easily guessable passwords.

For authentication mechanism 3: VPN-IPsec, the vulnerability can be attributed to implementation errors. To counter dictionary attack, the protocol obfuscates the group ID. However, the Contivity client software makes the effort useless by displaying the group ID in clear. In addition, the Contivity client software greatly facilitates the effort for a dictionary attack by displaying the number of characters used in the group password.

It is likely that only highly security-conscious users will use VPN-IPsec for authentication presently, which will not be too many. In view that the group ID and password will be used for long-term basis, UBC Wireless project may wish to evaluate alternative VPN-IPsec client software for replacement. Alternatively, instead of using group ID and password for authentication in Phase 1, other mechanism like the use of certificate may be considered.

While discussing with J. Martell, we were told that a fourth type of authentication mechanism, viz. 802.1x [16], is available for trial, although it is not made known publicly. We did not proceed with analysis and experiment due to time constraint, and also because it is not included in our project proposal. Hopefully this fourth authentication mechanism will be secure so that it may be considered as a replacement to existing authentication mechanisms if the trial is successful.

Acknowledgment

We would like to thank J. Martell, UBC Wireless Project Manager, for his time and effort in meeting and explaining to us the implementation of the UBC wireless network.

References

[1] Airsnort website: http://airsnort.shmoo.com/

[2] S. Beaulieu, and R. Pereira, "Extended Authentication within IKE (XAUTH)," Internet Draft (expired), Oct 2001. Available: http://www.vpnc.org/ietf-xauth/draft-beaulieu-ike-xauth-02.txt

[3] E. Biham, "A Fast New DES Implementation in Software," Proceedings of the 4th International Workshop on Fast Software Encryption, LNCS Vol. 1267, 1997.

[4] N. Borisov, I. Goldberg, and D. Wagner, “Intercepting Mobile Communications: The Insecurity of 802.11”, 7th Annual International Conference on Mobile Computing and Networking (MOBICOM), 2001.

[5] W. Diffie, and M. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, Vol. 22, 1976.

[6] W. Diffie, and M. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard," IEEE Computer, Vol. 10, 1977.

[7] distributed.net website: http://www.distributed.net/des/

[8] Electronic Frontier Foundation, "Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design," O'Reilly & Associates Inc., May 1998.

[9] Ethereal website: http://www.ethereal.com/

[10] N. Ferguson, and B. Schneier , "A Cryptographic Evaluation of IPsec", Apr 1999. Available: http://www.schneier.com/paper-ipsec.html

[11] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4", in Proceedings of the 8th Annual International Workshop on Selected Areas in Cryptography, LNCS Vol. 2259, 2001.
[12] A. Frieier, P. Karlton, and P. Kocher, "The SSL Protocol Version 3.0," Internet Draft (expired), Mar 1996. Available: http://wp.netscape.com/eng/ssl3/ssl-toc.html
[13] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn, "Point-to-Point Tunneling Protocol (PPTP)," RFC 2637, Jul 1999.
[14] D. Harkins, and D. Carrel, "The Internet Key Exchange," RFC 2409, Nov 1998.
[15] IEEE, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," ANSI/IEEE Std 802.11, 1999.
[16] IEEE, "Port-Based Network Access Control," ANSI/IEEE Std 802.1X, 2001.
[17] S. Kent, and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, Nov 1998
[18] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104, Feb 1997.
[19] L0phtCrack tool. Available: http://www.packetstormsecurity.org/Crackers/NT/l0phtcrack/
[20] S. Mamros, "Pre-Shared Key Extensions for ISAKMP/Oakley," Internet Draft (expired), Nov 1997. Available: http://hegel.ittc.ukans.edu/topics/internet/internet-drafts/draft-m/draft-mamros-pskeyext-00.txt
[21] J. Martell, D. G. Michelson, S. G. Mair, and D. Zollmann, "Deployment of Canada's Largest Campus Wireless Network at the University of British Columbia," Proceedings of International Conference on Information Technology: Research and Education (ITRE), Aug 2003.
[22] D. Maughan, M. Schertler, M. Schneider, and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)," RFC 2408, Nov 1998.
[23] NIST, "Secure Hash Standard," FIPS PUB 180-1, May 1993.
[24] NIST, "Data Encryption Standard (DES)," FIPS PUB 46-3, Oct 1999.
[25] G. Pall, G. Zorn, "Microsoft Point-to-Point Encryption (MPPE) Protocol," RFC 3078, Mar 2001.
[26] R. Perlman, and C. Kaufman, "Analysis of the IPSec Key Exchange Standard", in Proceedings of the 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE), Jun 2001.
[27] I. Petre course website: http://www.abo.fi/~ipetre/crypto/
[28] R. Rivest, "The MD4 Message Digest Algorithm," in Proceedings of Advances in Cryptology - CRYPTO '90, LNCS Vol. 537, 1991.
[29] R. Rivest, "The MD5 Message Digest-Algorithm," RFC 1321, Apr 1992.
[30] B. Schneier, and Mudge, "Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP)", in Proceedings of the 5th ACM Conference on Communications and Computer Security, Nov 1998.
[31] B. Schneier, Mudge, and D. Wagner, "Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2)", Secure Networking - CQRE '99, LNCS Vol. 1740, 1999.
[32] SMAC website: http://www.klcconsulting.net/smac/
[33] A. Stubblefield, J. Loannidis, and A. Rubin, "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP", in Proceedings of the Symposium on Network and Distributed System Security, 2002.
[34] UBC Wireless website: http://www.wireless.ubc.ca/
[35] D. Wagner, and B. Schneier, "Analysis of the SSL 3.0 Protocol", in Proceedings of the 2nd USENIX Workshop on Electronic Commerce, Nov 1996.
[36] J. Yan, " A Note on Proactive Password Checking," Proceedings of ACM Workshop on New Security Paradigms, Sep 2001.
[37] G. Zorn, "Microsoft PPP CHAP Extensions, Version 2," RFC 2759, Jan 2000.
[38] G. Zorn, "Deriving Keys for use with Microsoft Point-to-Point Encryption (MPPE)," RFC 3079, Mar 2001.