Wireless (In-)Security

Frank Kargl
frank.kargl@ulm.ccc.de

slides partially based on material from
Nikita Borisov at "Mobile Computing and Networking 2001"
used with permission of author

Overview

- Motivation & Introduction WEP
- Chronology of Cryptoanalysis of WEP
  - Oct. 2000 Walker
    "Unsafe at any key size; An Analysis of the WEP encapsulation"
  - Jan. 2001 UCB Goldberg, Borisov, Wagner
  - March/May 2001 Arbaugh, Shankar, Wan
    "Your 802.11 network has no clothes"
  - June 2001 Tim Newsham
  - Aug. 2001 Fluhrer, Mantin, Shamir
    "Weaknesses in the Key Scheduling Algorithm of RC4"
  - Aug. 2001 Stubblefield, Ionnidis, Rubin
    "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP"
  - Aug. 2001 WEPcrack/AirSnort
- Drive-By Hacking/War Driving
- Securing Wireless LANs
- Will WEP Version 2 be better ???

Questions

- Who owns a 802.11 NIC?
- Who has connected it to an open foreign network?

Wireless Security

- Wireless networks becoming prevalent
- New security concerns
  - More opportunities for attack
  - Possible to monitor and participate in a network at a distance
    (1/2 mi or even further)
- The 802.11 answer: WEP
  - "Wired Equivalent Privacy"

WEP Security Goals

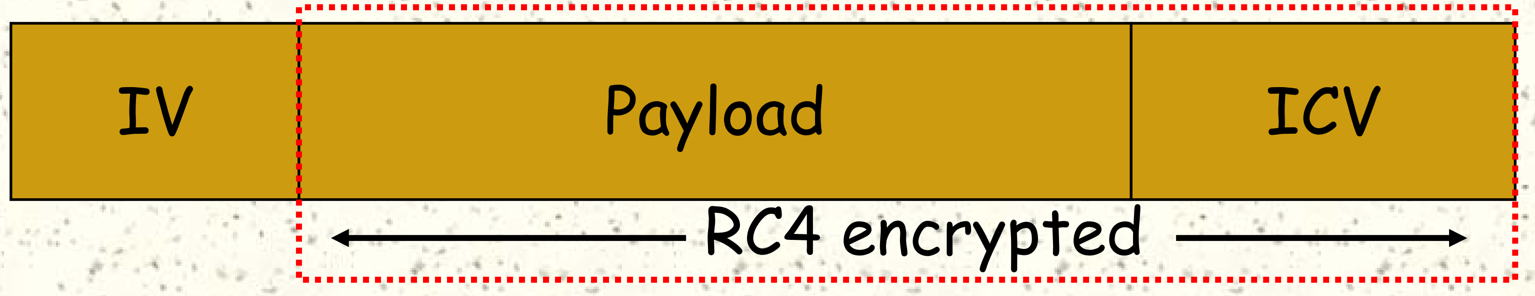
- Prevent link-layer eavesdropping
  - … not end-to-end security
- Protect message integrity
- Control network access
- Essentially, equivalent to wired access point security
- None of these goals are met

Goldberg, Borisov, Wagner (UCB)

- First results published in the Internet January 2001
- Number of reviewed publications in
  MAC Crypto Workshop 2001
  Black Hat Briefing 2001
  Mobile Computing and Networking 2001
- Find additional information on
  http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html

Protocol Overview

- Mobile station shares key with access point
- Integrity check value computed on payload
- Payload + ICV are encrypted with the shared key & an initialization vector (IV)
- IV included in the clear
- Receiver decrypts, verifies ICV, and rejects packet if check fails



Encryption Algorithm

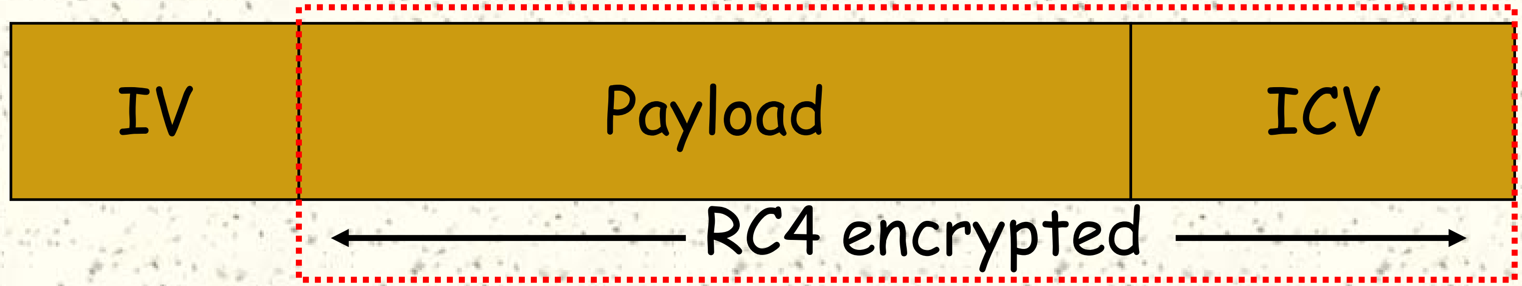
- RC4 – a well-studied algorithm
- RC4 is a stream cipher
- Expands a key into an infinite pseudorandom keystream
- To encrypt, XOR keystream with plaintext
- $Random \oplus Anything = Random$
- Encryption same as decryption

Keystream Reuse

- Using same part of RC4 keystream to encrypt two messages is disastrous:
  - $C1 = P1 \oplus RC4(key)$
  - $C2 = P2 \oplus RC4(key)$
  - $C1 \oplus C2 = P1 \oplus P2$
- Knowledge of P1 reveals P2
- More sophisticated analysis possible based on expected distribution of P1 and P2

Initialization Vectors

- Use initialization vectors to generate different keystream for each packet
- IV augments the shared key:
- $C = P \oplus RC4(key, IV)$
- Different IVs $\Rightarrow$ Different keystreams
- Include IV unencrypted in the packet



Problem 1: IV collisions, short IV

- IV *collisions* – two packets with same IV
  - Therefore same keystream
- 802.11 does not specify how to pick IVs
  - Doesn't even require a new one for each packet!
- Many implementations reset IV to 0 when initialized, increment with each packet
- Easy to find IV collisions!
  - Especially if shared key is used in both directions or by many mobile stations

IV collisions, continued

- 24-bit IV – 2^{24} possibilities
- Guaranteed collisions after sufficient time (a few hours to a few days)
- Known plaintext for one packet allows to decrypt any others with the same IV
  - Obtain 2^{24} known plaintexts
  - Store decryption table on a cheap hard drive
  - Ways to obtain plaintext: IP headers, login procedures, binaries being transfered, send SPAM via Email, Webpages ...
- **Confidentiality compromised**

Problem 2: Linear Checksum

- Encrypted CRC-32 used as integrity check
  - Fine for random errors, but not deliberate ones
- CRC is linear
- I.e. $CRC(X \oplus Y) = CRC(X) \oplus CRC(Y)$
- $RC4(k, X \oplus Y) = RC4(k, X) \oplus Y$
- $RC4(k, CRC(X \oplus Y)) = RC4(k, CRC(X)) \oplus CRC(Y)$
  - Hence we can change bits in the packet

Packet Modification

	Payload	CRC-32
	011010010100……………………………	10110…………
RC4	101101110101………………………………………………	
XOR		
	110111100001……………………………	11011…………
	010000000000……………………………	00110…………
XOR		
	100111100001……………………………	11101…………

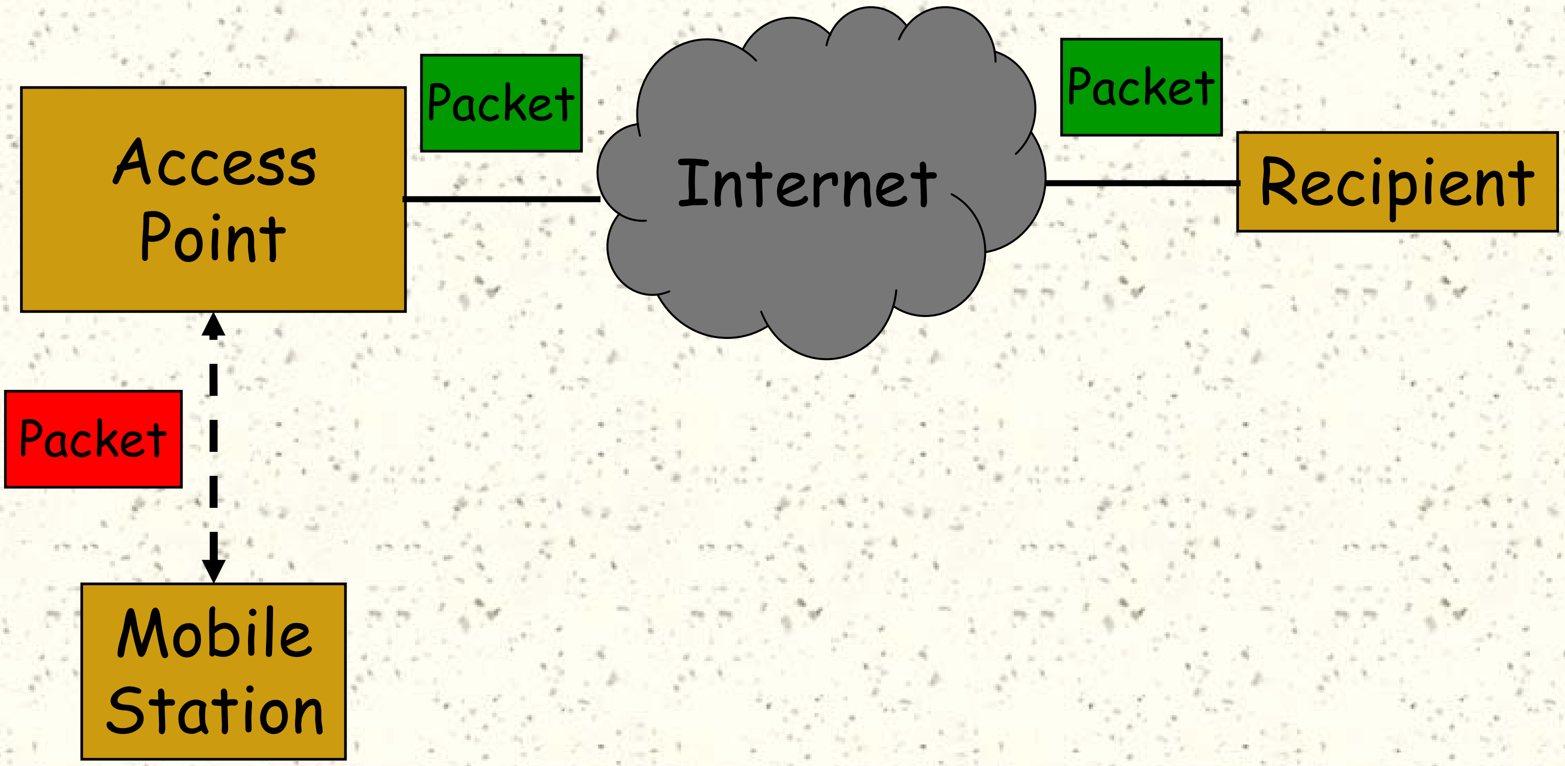
Modified Packet

$$RC4(k,CRC(X \oplus Y)) = RC4(k,CRC(X)) \oplus CRC(Y)$$

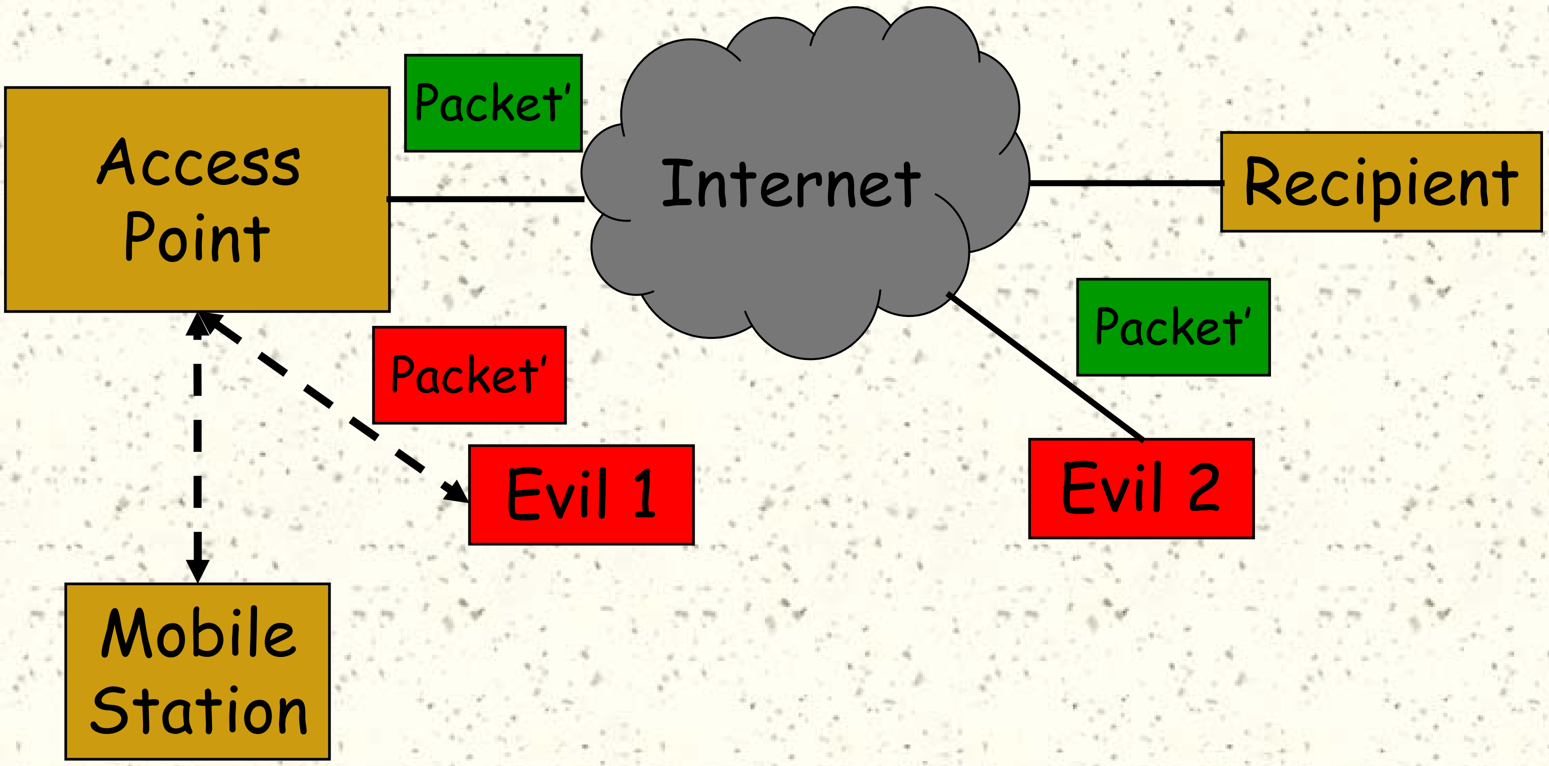
Can modify packets!

- "Integrity check" does not prevent packet modification
- Can maliciously flip bits in packets
  - Modify active streams!
  - Bypass access control
- Partial knowledge of packet is sufficient

Typical Operation



Redirection Attack



Redirection Attack

- Suppose we can guess destination IP in encrypted packet
- Flip bits to change IP to Evil 2, send to AP
  - Tricks to adjust IP checksum (complementary changes ...)
- AP happily forwards it to Evil 2
- Set port to 80 to bypass firewalls
- Incorrect TCP checksum not checked until Evil 2 sees the packet!

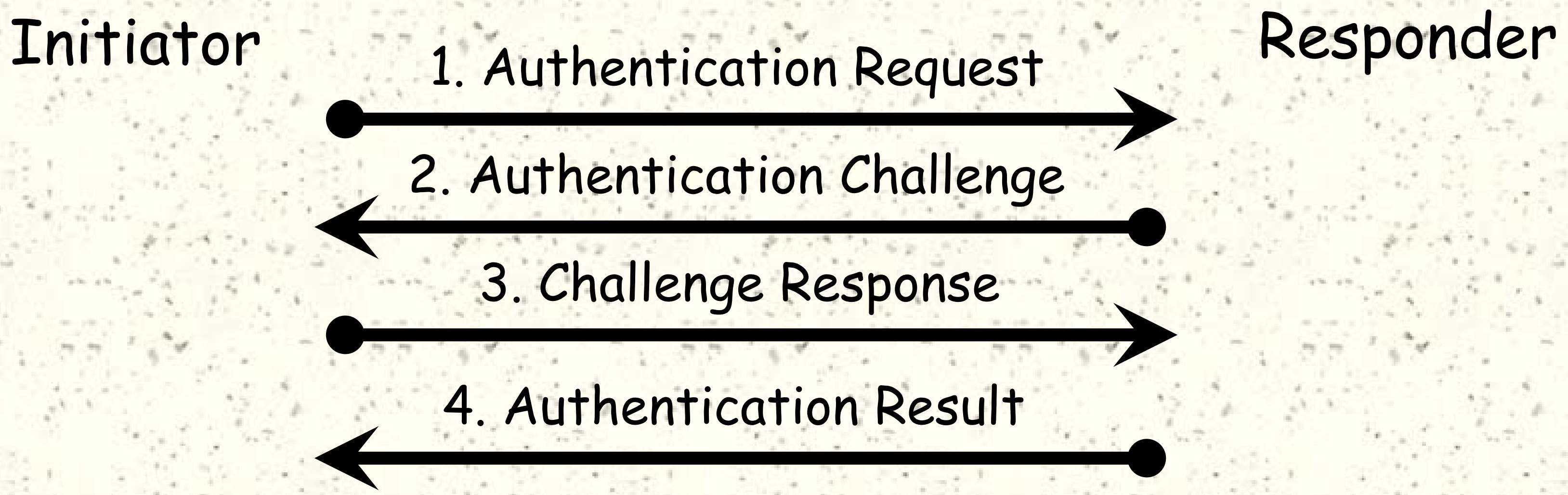
Message Integrity Essential

- Poor integrity checks can lead to compromised confidentiality, too
  - Redirection attack
  - TCP reaction attack
    - Use TCP checksum/ACK to recover plaintext
  - Inductive CRC attack (see later)
- Rule of thumb: whenever you encrypt, also use a MAC

Arbaugh, Shankar, Wan (College Park)

- Published March/May 2001
  "Your 802.11 network has no clothes"
- Followup May 2001
  "An Inductive Chosen Plaintext Attack Against WEP/WEP2"

Shared Key Authentication



1. Request
2. Challengetext
3. WEP(key, Challengetext)
4. ACK/NACK

Attacking Shared Key Auth.

1. Capture Message 2 and 3
   Known: Challengetext P,
   Ciphertext C,
   Complete Pseudo-Random Stream
   $RC4(K, IV) = C \oplus P$
2. Authenticate
   Get new Challengetext P'
   Construct Reply using
   $P' \oplus RC4(K, IV)$

Inductive Chosen Plaintext Attack

- Problem with IV Berkeley Attack: Must somehow know plaintext
- Idea: when knowing pseudo-random stream of length n, extend to n+1 by using information from ICV

Inductive Chosen Plaintext Attack

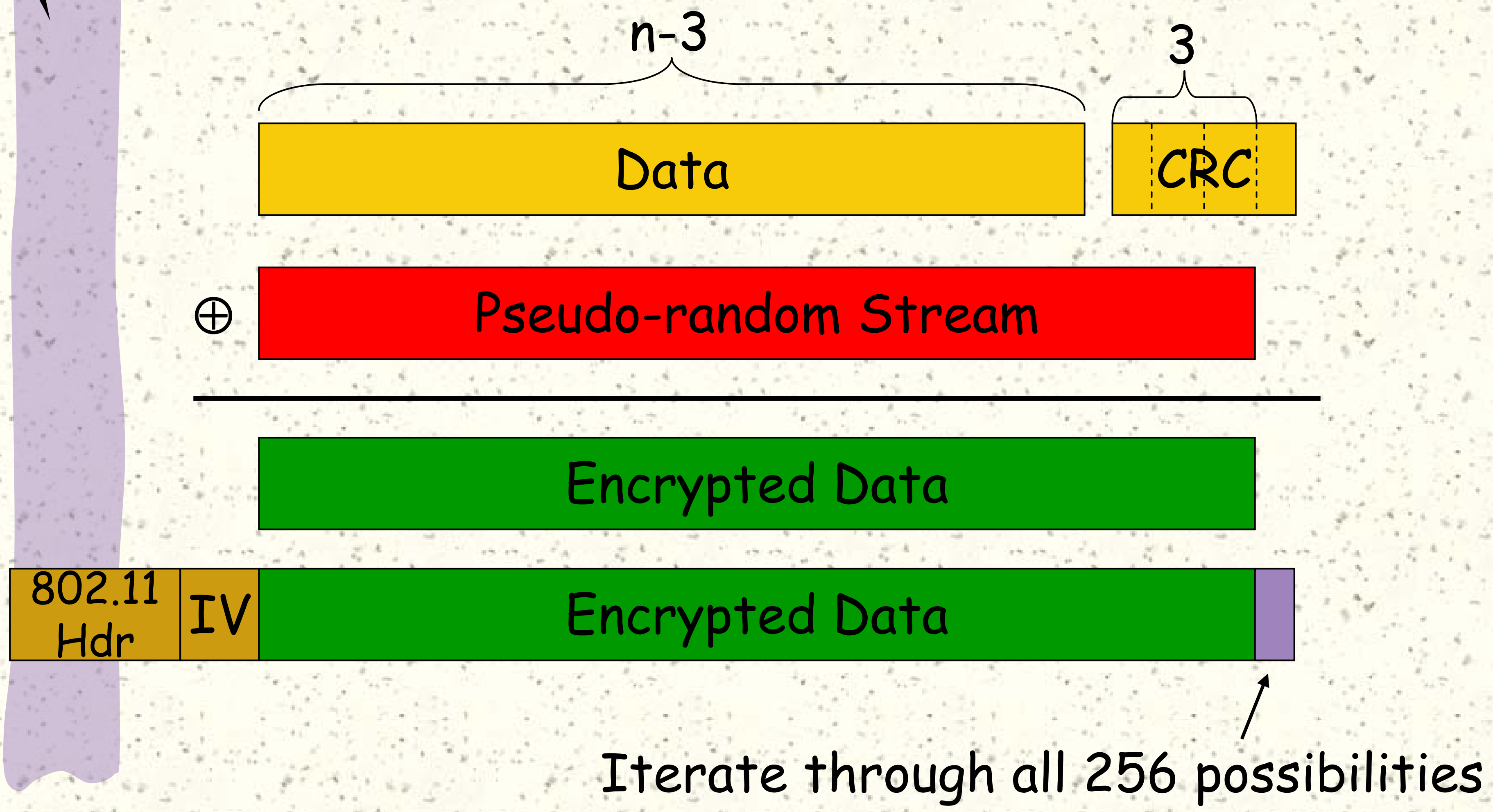
- Start:
  Get start of pseudo-random stream from known packet
- E.g. start of DHCP request
  (Source IP: 0.0.0.0,
  Dest. IP: 255.255.255.255)
- Allows the recovery of first 24 byte of pseudo-random stream, n=24

Inductive Chosen Plaintext Attack

Inductive step:

1. Generate Datagram of size n-3 (e.g. ARP Request etc.)
2. Compute CRC, use only 3 bytes
3. XOR with n bytes pseudo-random stream
4. Append last byte as n+1

Inductive Chosen Plaintext Attack

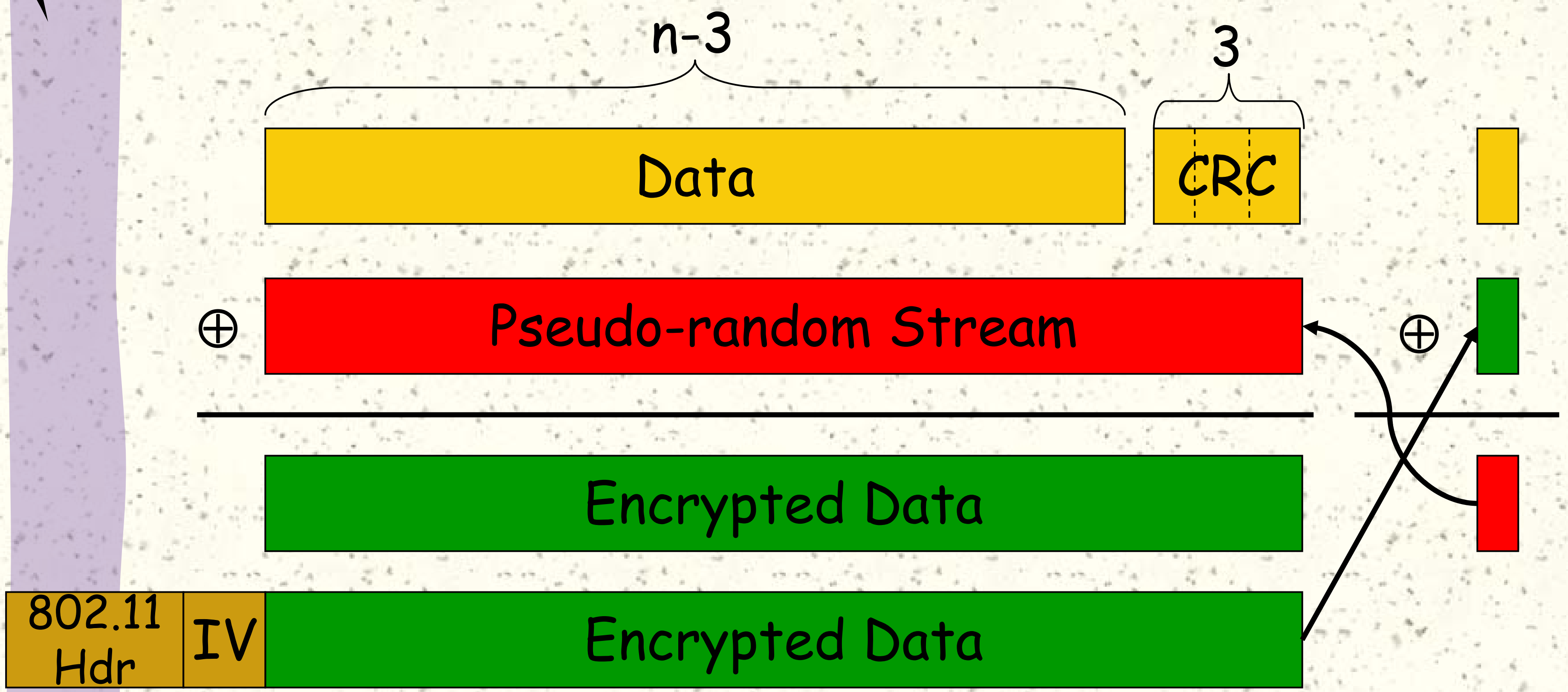


Inductive Chosen Plaintext Attack

Inductive step:

5. Now send datagram and wait for response
6. If no response, try next
7. If there is a response, we know:
   The n+1 byte was the correct CRC, so we have a matching plain- and ciphertext, giving us a pseudo-random stream of length n+1

Inductive Chosen Plaintext Attack



Attack Cost

- Assume ~ 100 Packets / s
  - Attack difficult to observe, as CRC errors are filtered by hardware;
    CRC failure counters
  - ~1h to recover 1500 byte MTU
    (regardless of key and IV size; WEP2)
  - ~30 minutes in average case
  - ~1000 years for full 24bit dictionary
  - but blindly sending packets or listening to some packets may be enough

Tim Newsham

- June 2001: "Cracking WEP Keys"
- ASCII key generators badly flawed
- Entropy goes down
  64 bit generator: 21 bit entropy
  128 bit generator: depends on length of passphrase
- Implemented dictionary and brute force attack
- Sourcecode is PD

That's it ?

- No, the fun is just beginning

Fluhrer, Mantin, Shamir

- Published Aug. 2001
  "Weaknesses in the Key Scheduling Algorithm of RC4"
- Shamir one of the inventors of RC4

RC4 Operation

- KSA(K)

Initialization:

For i=0...N-1

    S[i]=i

j=0

Scrambling:

For i=0...N-1

    j=j+S[i]+K[i mod l]

    Swap(S[i],S[j])

- PRGA

Initialization:

i=0

j=0

Generation Loop:

    i=i+1

    j=j+S[i]

    Swap(S[i],S[j])

    Output z=S[S[i]+S[j]]

Result 1: The Invariance Weakness

- KSA and PRGA are highly biased for certain keys
- Weak Key Patterns -> Permutation -> Output
- Applications:
  - Highly efficient distinguishers
  - Time/memory/data tradeoff attacks (RC4 has Low Sampling Resistance)

Result 2: Known IV Attack

- For some special IVs, knowledge of the IV plus the first cleartext byte (often 0x04 or 0xaa) reveals information about the secret key (with a probability of > 0.05)
- Solution: Analyze enough packets, so you can reconstruct the key with high confidence

Stubblefield, Ionnidis, Rubin

- Aug. 2001 published
  "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP"
- Implemented and refined the idea of Fluhrer, Mantin and Shamir within about 2 weeks
- Were able to find the key of their 802.11 network (128 bit) within a few hours
- Needed about 5.000.000 packets collected
- Didn't publish their software

WEPcrack/AirSnort

- Open Source projects hosted at Sourceforge
- Needs PrismII based cards
- Implements the former attack
- Ideal case: 15 minutes
- Otherwise: up to a few days

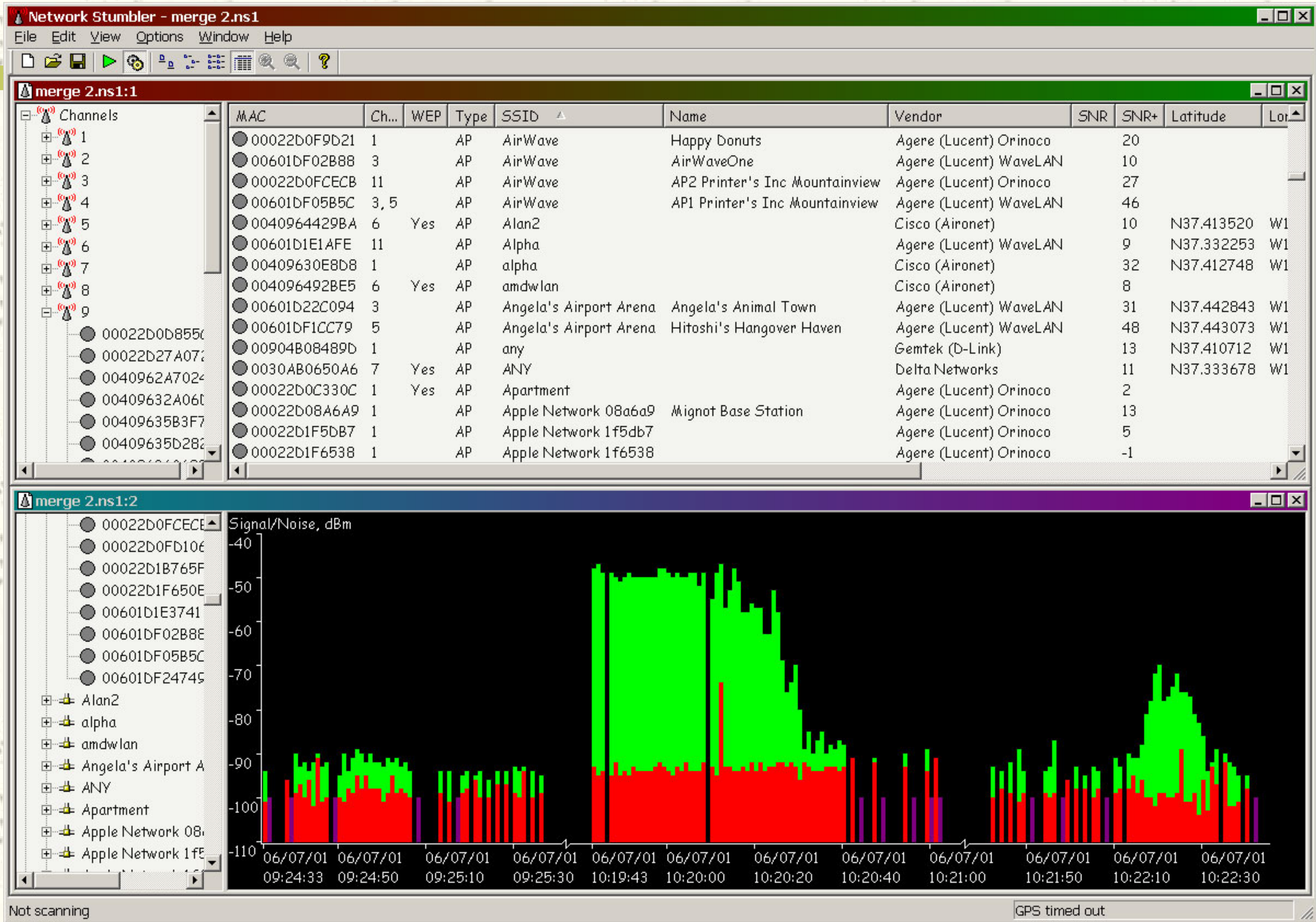
Attack Practicality

- Sit outside competitor's office, use a notebook and an off the shelf wireless card
- With minimal work, possible to monitor encrypted traffic
  - PrismII based cards + prismdump + ethereal
  - Commercial sniffers also available
- Software for reconstructing the key readily available

Drive-By Hacking

- Sometimes also called War-Driving
- Bay Area Wireless User Group

Netstumbler



Lessons

- Security hard to achieve
  - Even when good crypto is used
- Conflicting design objectives
  - E.g. "be liberal in what you accept"
- Public review is a Good Idea
  - Time to develop attacks is short!
  - Reaction: 802 standards are now freely available, WEP2 is developed in an open process
- Use previous work (and their failures)
  - Similar attacks on other systems (e.g. earlier versions of IPSEC) have been shown

Conclusion

- Demonstrated problems in WEP arising from misuse of cryptographic primitives or weaknesses in the cryptographic ciphers themselves
- Attacks compromising all of the security goals of WEP

Security Measures

- Treat 802.11 networks as untrusted
- Plase 802.11 networks outside of your firewall (or in the DMZ; like Dialin)
- Use WEP, but don't trust it alone
- Use every mechanism your 802.11 system offers (e.g. Lucent/Orinoco hidden networks)
- Use additional technologies (e.g. VPN, end-to-end encryption) to secure transmissions

Filter on MACs?

Forget it:

```
// Make the MAC address valid by:
//   - Clearing the multicast bit
//   - Setting the local MAC address bit
lp->MACAddress[0] &= ~0x03;
lp->MACAddress[0] |= 0x02;
```

WEP2

- Currently in Progress
- New Services: key distribution, data authentication, replay protection
- Still RC4, but will use 128 bit IVs and longer keys (256 bit)
- Some of the attacks are independent of the IV and key size or even get easier with longer IVs
- Hard to implement as firmware on current ASICs -> new cards?

Ressources

- Berkeley
  http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html
- College Park
  http://www.cs.umd.edu/~waa/wireless.html
- Tim Newsham
  http://www.lava.net/~newsham/wlan/
- Stubblefield et.al.
  http://www.cs.rice.edu/~astubble/wep/
- Fluhrer et.al.
  e.g. http://www.crypto.com/papers/others/rc4_ksaproc.ps

Software

- AirSnort
  http://airsnort.sourceforge.net/
- WEPcrack
  http://wepcrack.sourceforge.net/
- netstumbler
  http://www.netstumbler.com/

The End

Questions and Answers