Analysis of 802.11 Security *or Wired Equivalent Privacy* ***Isn't***

Nikita Borisov, Ian Goldberg,
and David Wagner

WEP Protocol

- “Wired Equivalent Privacy”
- Part of the 802.11
- Link-layer security protocol

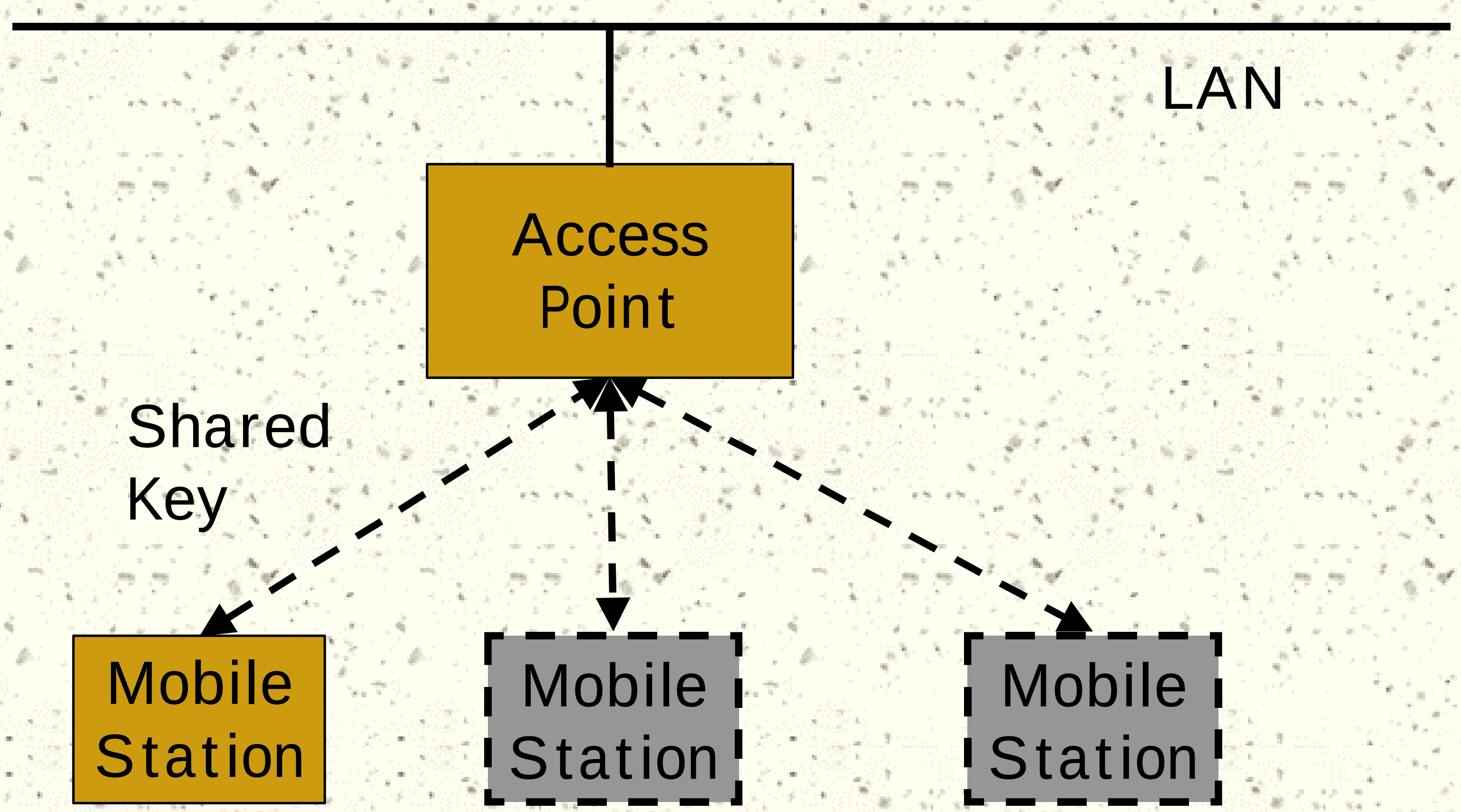
Security Goals

- Prevent link-layer eavesdropping
  - ... not end-to-end security
- Secondary goal: control network access
  - Not always an explicit goal
- Essentially, equivalent to wired access point security

"Open Design"?

- An industry-driven committee (?)
- No apparent public review (**X**)
- Resulting standard is open ... ( )
- ... but costs $$$ (**X**)
- Use a well-studied cipher ( )

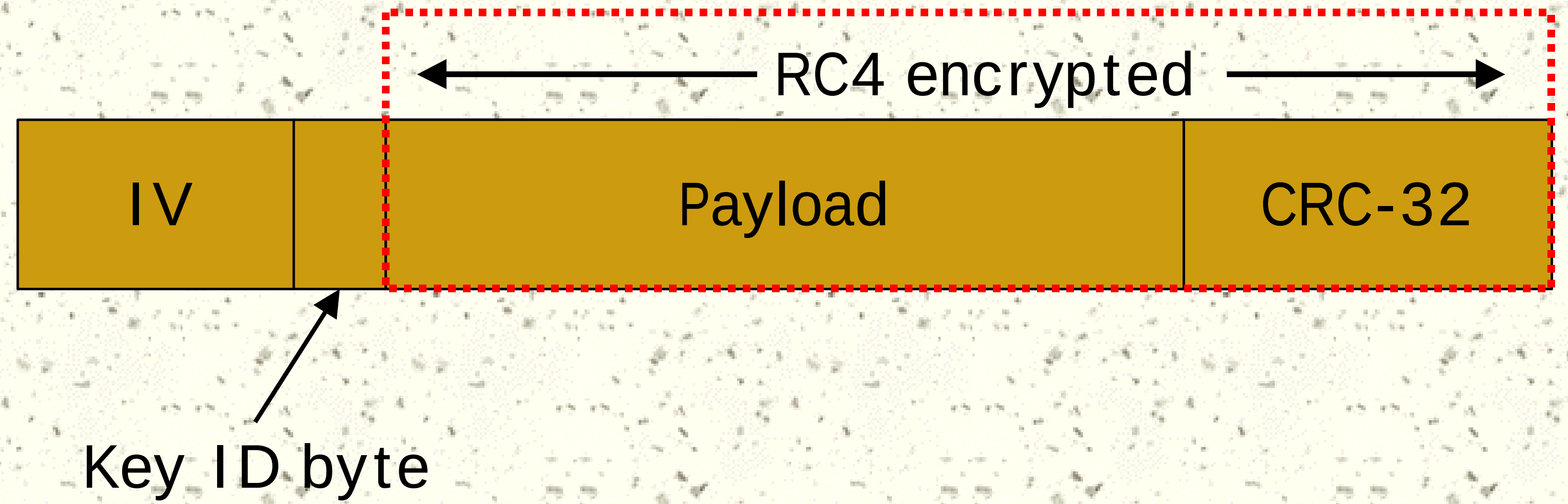
Protocol Setup



Protocol Setup

- Mobile station shares key with access point
- Each packet is encrypted with shared key + initialization vector (IV)
- Each packet includes an integrity check
- IC fails => packet rejected
- Optionally, reject all unencrypted packets

Packet Format



Problem 1: Stateless Protocol

- Mobile stations and access points are not required to keep past state
- Fundamental consequence: can replay packets
- But IP allows for duplication anyway, right?

Stream Ciphers

- RC4 is a stream cipher
- Expands a key into an infinite pseudorandom keystream
- To encrypt, XOR keystream with plaintext
- *Random ^ Anything = Random*
- Encryption same as decryption

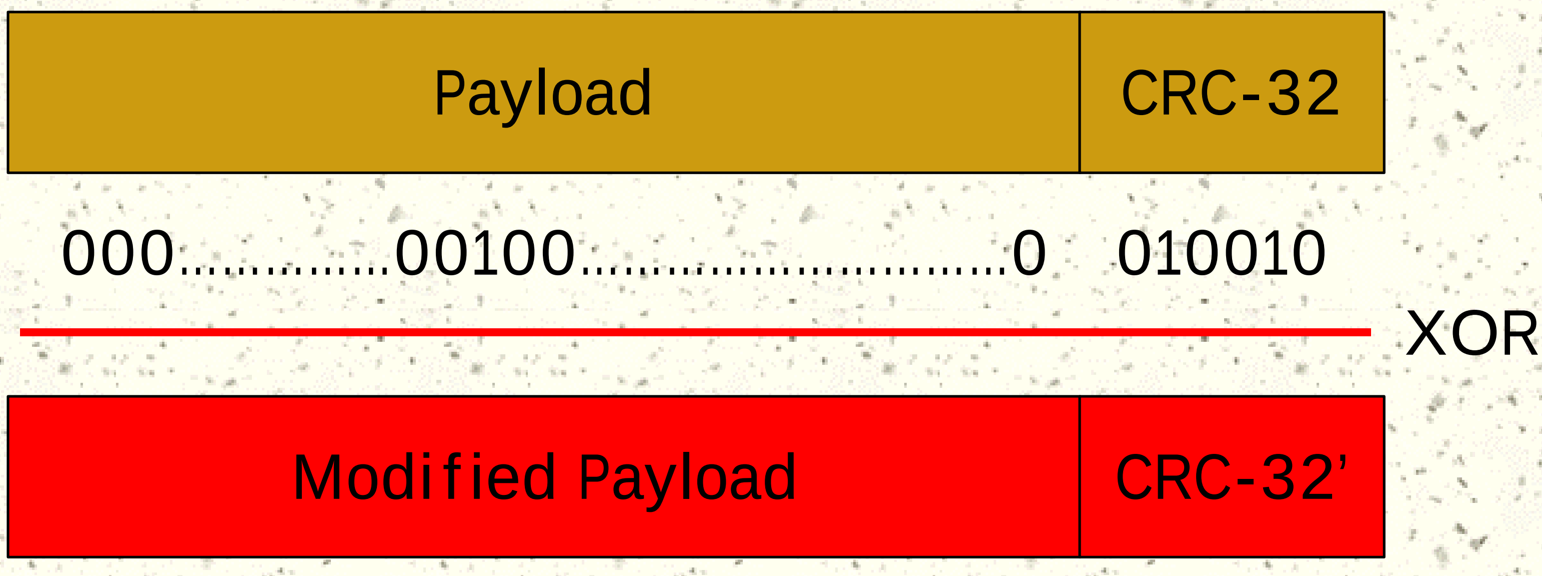
Example

“WIRELESS”	=	**584952454C455353**
RC4(“foo”)	=	**123456789ABCDEF**
		XOR
		4A7D043D6FBE9C
RC4(“foo”)	=	**123456789ABCDEF**
		XOR
“WIRELESS”	=	**584952454C455353**

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^Y) = CRC(X)^CRC(Y)
- RC4(k,X^Y) = RC4(k,X)^Y
  - Hence we can change bits in the packet

Packet Modification



Can replay modified packets!

- "Integrity check" does not prevent packet modification
- Can maliciously flip bits in packets
  - Modify active streams!
- TCP checksum: not quite linear, but can guess right about half the time
- Known plaintext for a single packet allows to send arbitrary traffic!

What about IVs?

- RC4 keystream should not be reused, since RC4(k,X)^RC4(k,Y) = X^Y
- Use initialization vector to generate different keystream for each packet by augmenting the key
- Key = base_key || IV
- Include IV (plaintext) in header

Problem 3: IV reuse

- Same shared key used in both directions
  - … on some installations all stations share same key
  - I.e. a “network password”
- Some implementations reset IV to 0 when initialized
- Easy to find collisions!

IV collision

- Two packets P1 and P2 with same IV
- C1 = P1 xor RC4(k||IV)
- C2 = P2 xor RC4(k||IV)
- C1 xor C2 = P1 xor P2
- Known plaintext P1 gives P2, or use statistical analysis to find P1 and P2
- Even easier if you have three packets!

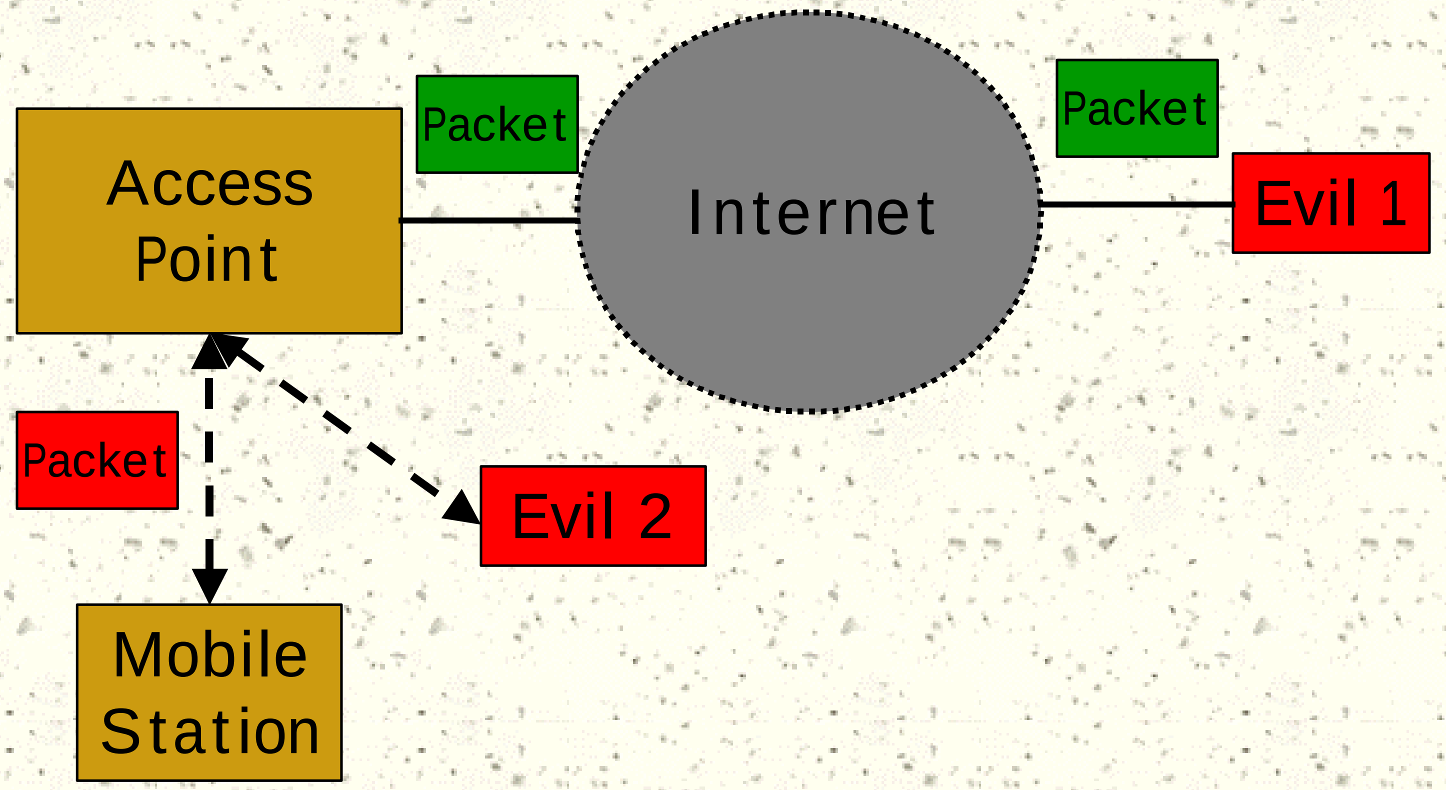
Implementation bug or design flaw?

- What if random IVs were used?
- IV space – 2^{24} possibilities
- Collision after 4000 packets
- Rough estimate: a busy AP sends 1000 packets/sec
- Collision every 4s!
- Even with counting IV (best case), rollover every few hours

IV collisions, continued

- If we have 2^{24} known plaintexts, can decrypt every packet
  - Becomes more of a problem if stronger crypto (ie. 128-bit RC4) is deployed
- How to get known plaintext?
- IP traffic pretty predictable
- Authentication challenge?
- Send packets from outside?

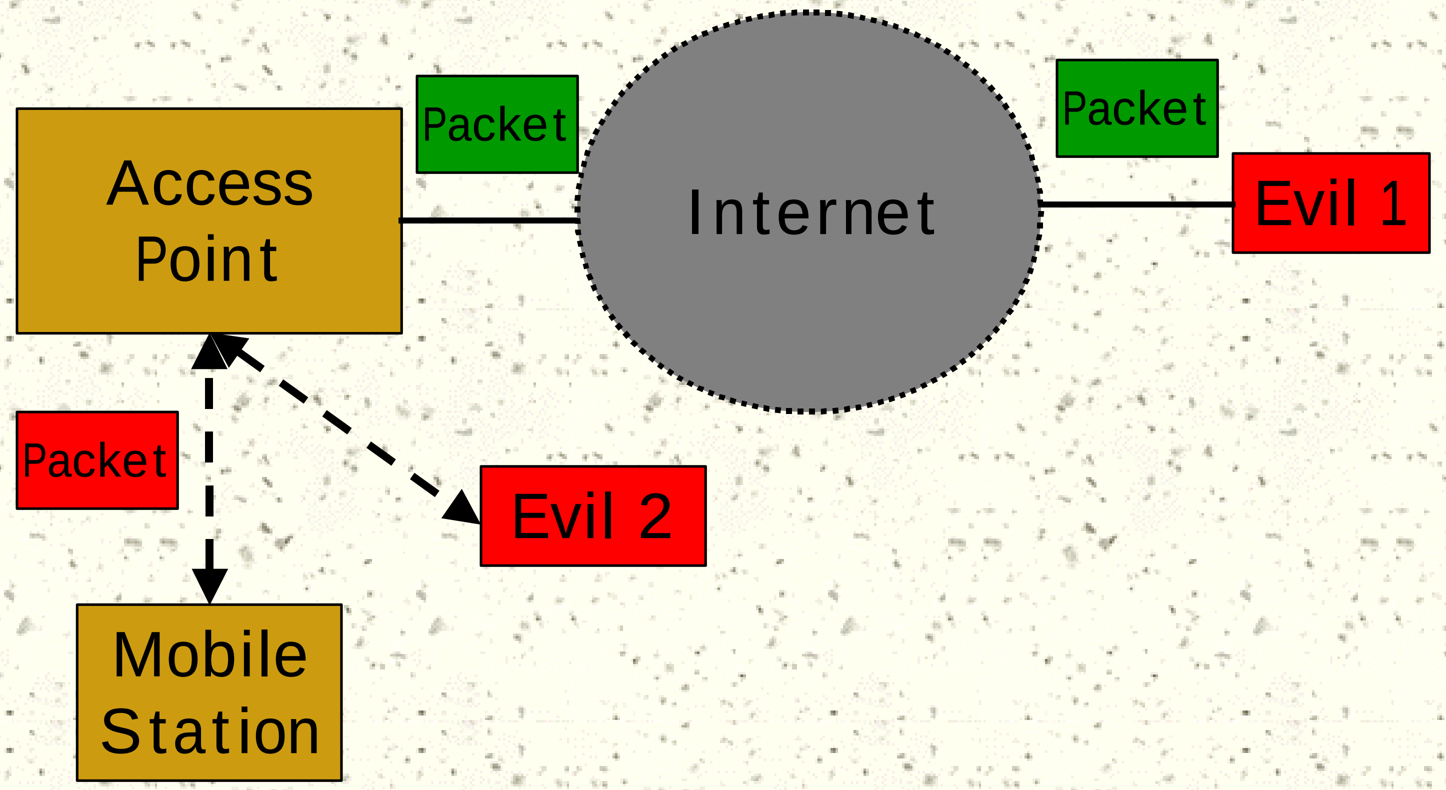
Attack from Both Ends



Problem 4: Encryption Oracle

- Access Points encrypts packets coming from the LAN before sending over air
- LAN eventually connects to Internet; attack AP from both ends
- Send packets from Internet with known content to a wireless node
- Voila! Known plaintext

Attack from Both Ends (2)



Decryption Oracle??

- Recall Problem 2: can flip bits in packets
- Suppose we can guess destination IP in encrypted packet
- Flip bits to change IP to host we control, send it to AP
  - Tricks to adjust IP checksum
- AP happily forwards it to the our host
- Set port 80 to bypass firewalls
- Incorrect TCP checksum not checked until we see the packet!

Attack Practicality

- Sit outside competitor's office, use a software radio
- … or an off the shelf wireless card!
- With minimal work, possible to monitor encrypted traffic
- Reverse engineer firmware for active attacks
- Economies of scale: only has to be done once!

Lessons Not Learned

- Most attacks are not new!
- Earlier versions of IPSEC had many similar problems (e.g. [Bel96])
  - Other attacks (e.g. reaction) applicable
- SSH (and many others) had CRC checksum problems
- Microsoft PPTP had RC4 directionality problems

Lessons to take away

- Protocol design is harder than it looks
- Can be circumvented at many points
- Public review is a Good Idea™
  - Time to develop attacks is short!
- Use previous work (and their failures)
  - Put wireless network **outside** firewall, run VPN to inside firewall
  - Better yet, use end-to-end encryption