

# Wireless LAN Security



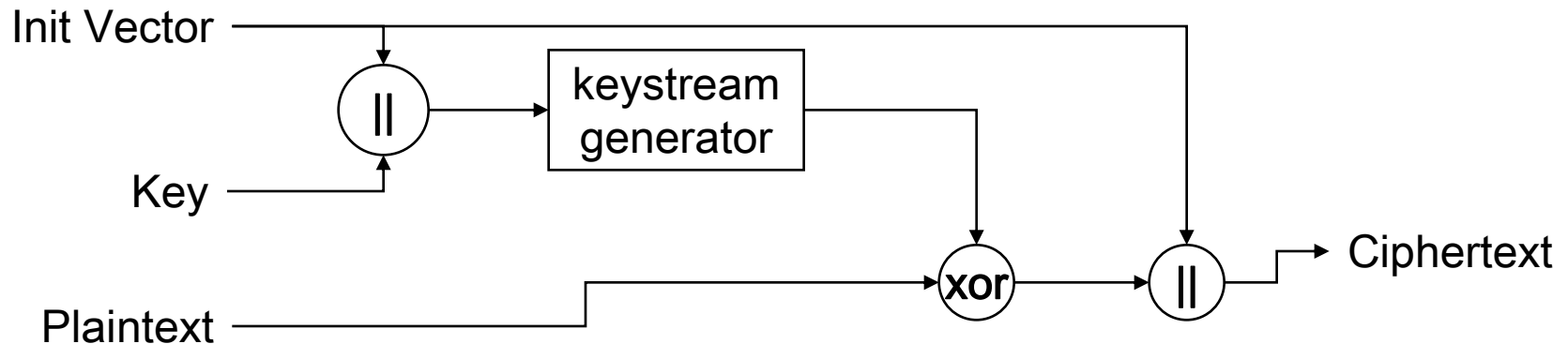
# WEP: Overview

- WEP = Wired Equivalency Protocol
- RC4 stream cipher
- Purposes:
  - Authentication
  - Packet Encryption
- Uses single key to authenticate all network users and encrypt all packets



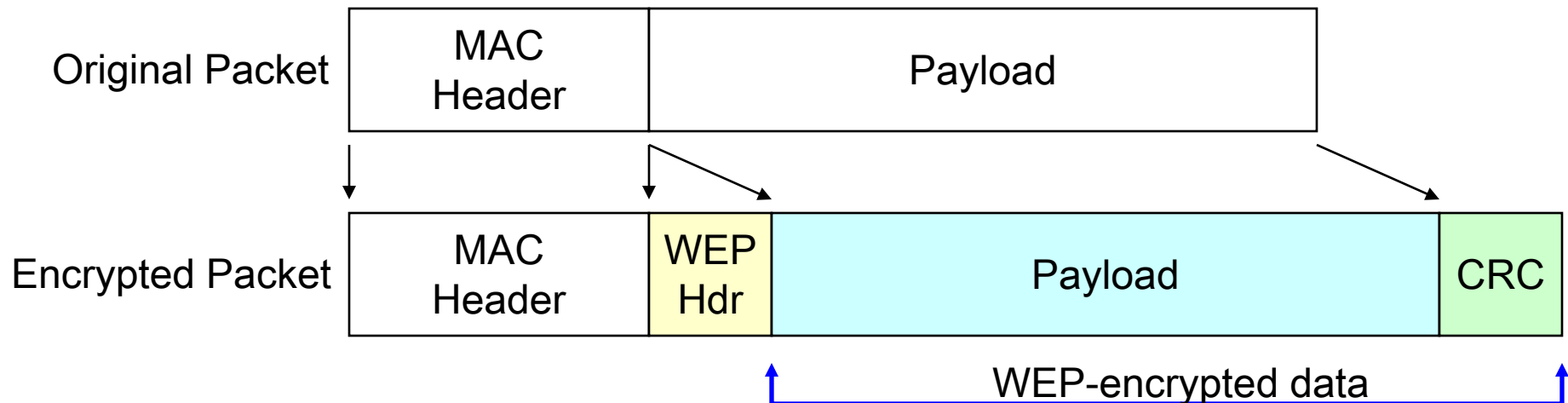
# WEP: RC4

- RC4 = Rivest Cipher #4
- Stream Cipher
- Same device for encryption and decryption because of the XOR



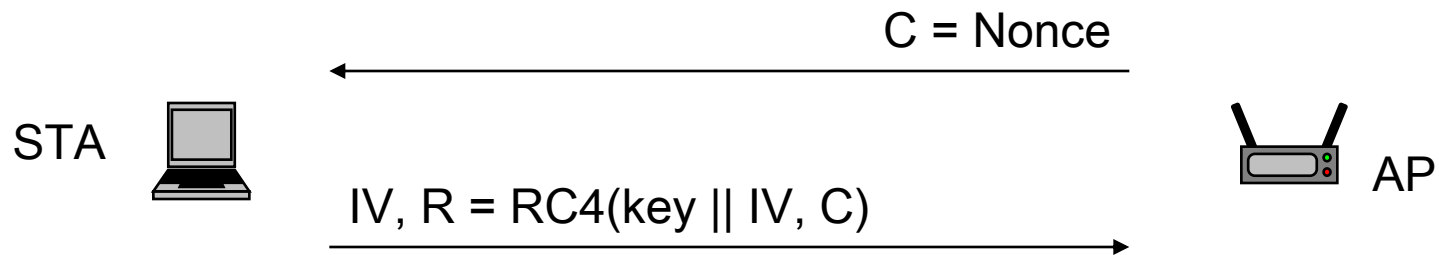
# WEP: Packet Format

- WEP-encrypted packets include an additional 4-byte header, and a 4-byte CRC-32
- WEP header includes the 24-bit IV and some flags
- CRC-32 covers *only* the payload, and is used to determine if a packet has been successfully decrypted



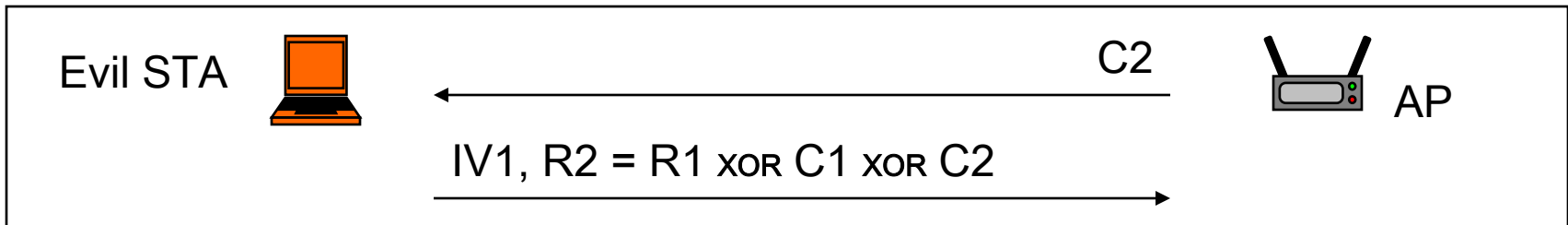
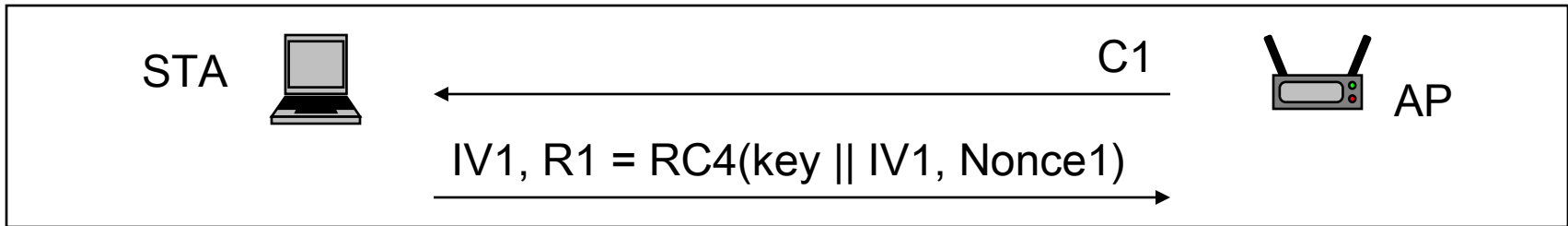
# WEP: Authentication

- WEP defines “shared-key authentication” for admission control to WEP-protected networks
- AP sends challenge C, STA WEP-encrypts it and responds with R



# WEP: Authentication

- Why is WEP shared-key authentication broken?



$$\begin{aligned} R2 &= R1 \text{ XOR } C1 \text{ XOR } C2 \\ &= (\text{keystream}(\text{key} \parallel \text{IV1}) \text{ XOR } C1) \text{ XOR } C1 \text{ XOR } C2 \\ &= \text{keystream}(\text{key} \parallel \text{IV1}) \text{ XOR } (C1 \text{ XOR } C1) \text{ XOR } C2 \\ &= \text{keystream}(\text{key} \parallel \text{IV1}) \text{ XOR } C2 \\ &= \textit{VALID RESPONSE} \end{aligned}$$



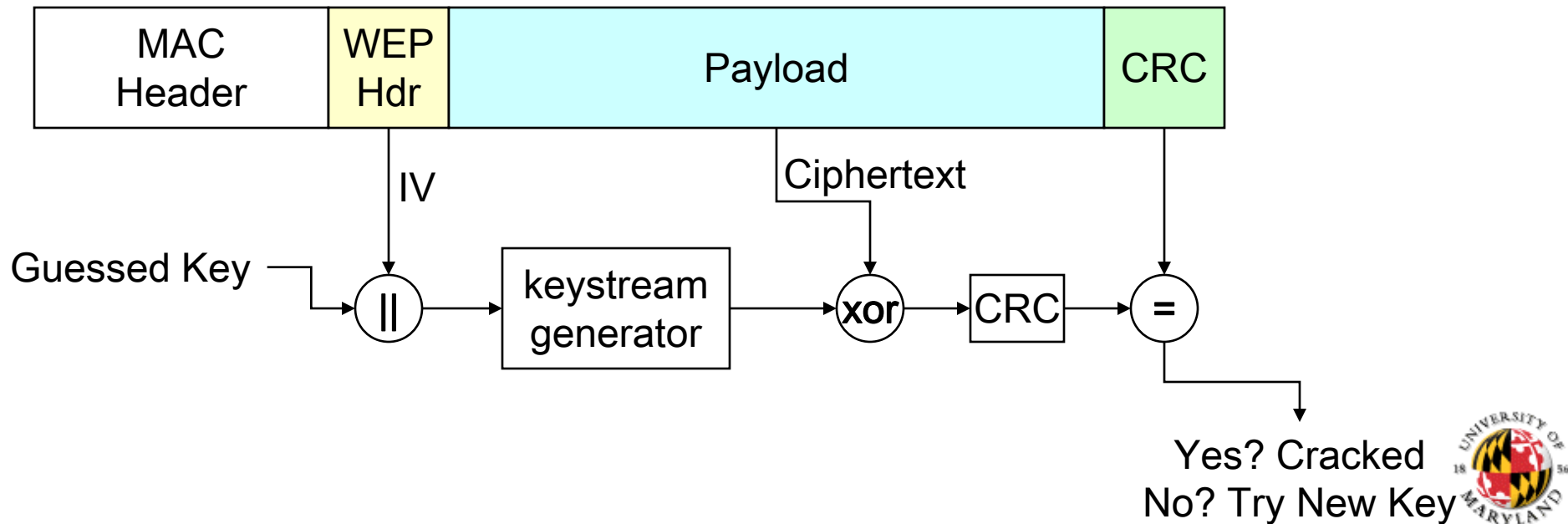
# WEP: Authentication

- **Problem:** Authentication Broken
- **Solution:** don't use it -- just encrypt packets
- Called “open authentication”
- You can associate to a network, but without the right key, the AP will drop all your packets



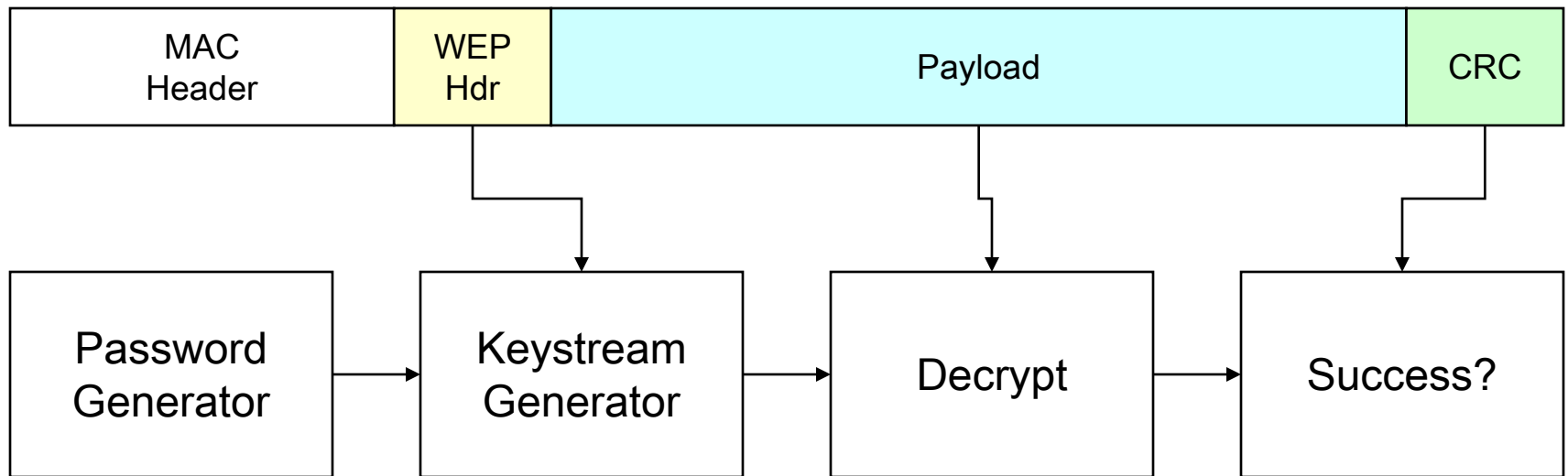
# WEP: Brute Force Attack

- Brute Force 40-bit or 104-bit key space
- COTS Hardware, 1 processor, 3GHz
  - 40-bit = 2 weeks
  - 104-bit =  $10^{18}$  years (90 years if you account for Moore's Law)



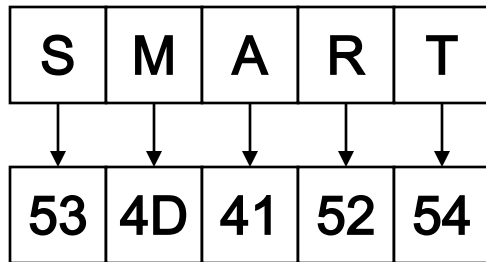
# WEP: Dictionary Attack

- Brute force the password space used for key generation, rather than the full key space
- Same speed for both 40-bit and 104-bit



# WEP: Dictionary Attack

- Key Generation
  - Direct ASCII translation

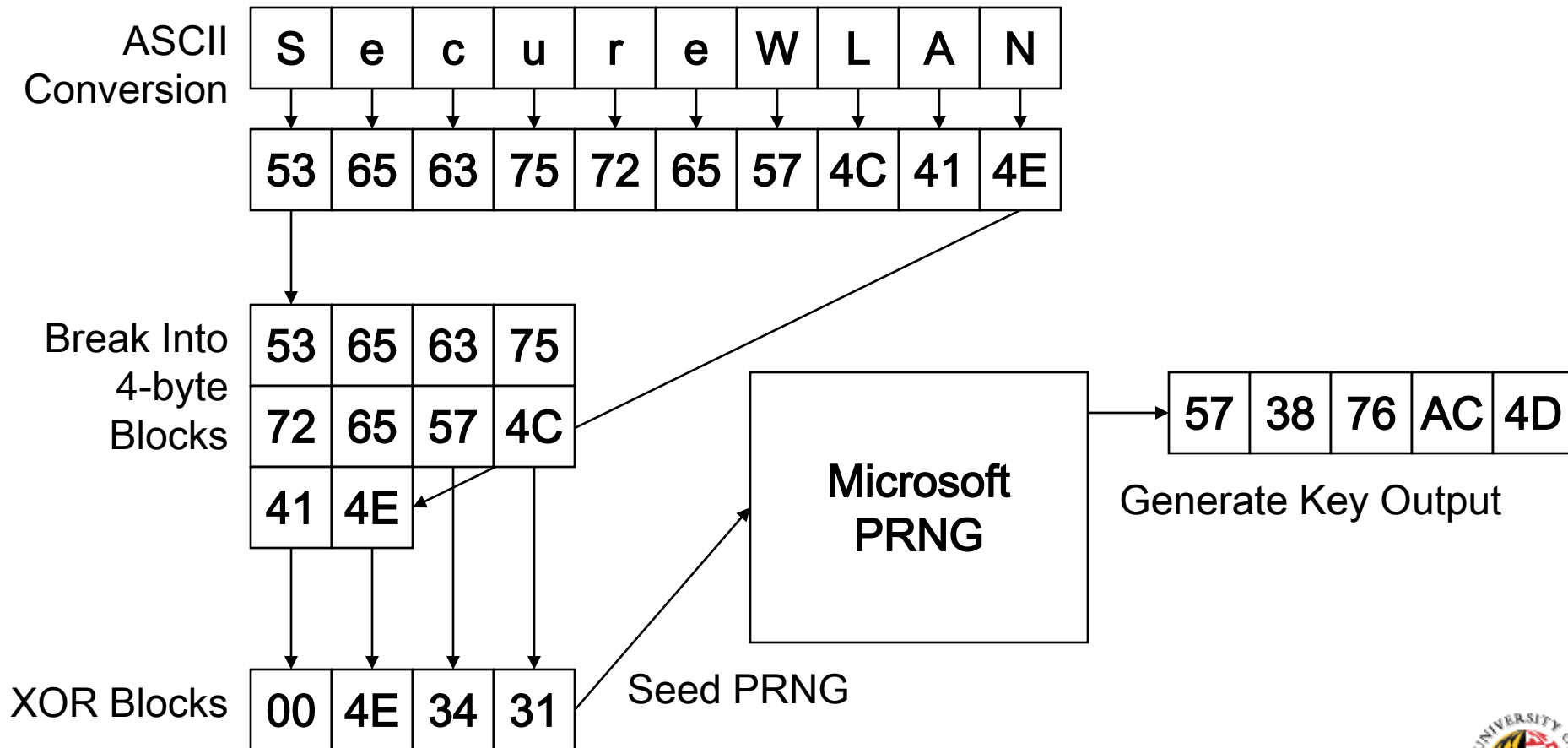


- MD5 Hashing



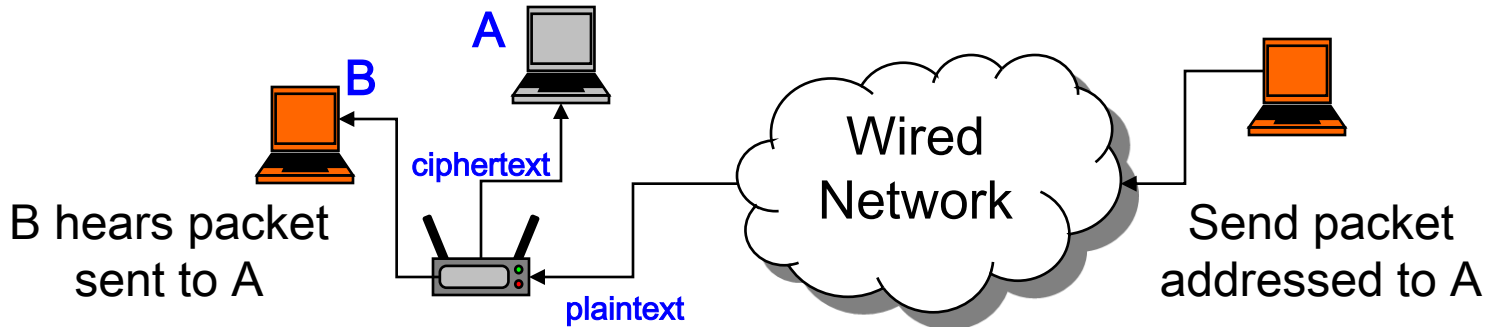
# WEP: Weak Key Generation Attack

- Microsoft / Linksys key generator is *by far the worst*



# WEP: Keystream Enumeration Attack

- Launch Chosen Plaintext Attack (CPA) against 24-bit IV space

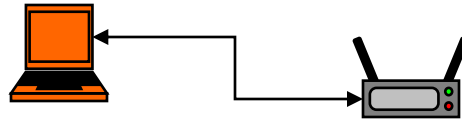


- Attacker XORs the plaintext and ciphertext to recover the keystream for the IV selected by the AP
- Build a database of keystreams for IVs
- Whenever you see an IV you have in your database, use the associated keystream to decrypt the packet



# WEP: Inductive Keystream Attack

- Launch Chosen Ciphertext Attack (CCA) to recover a keystream



- Inductively recover keystream using CRC attack
- Inductive Step: know N bytes of keystream, determine N+1

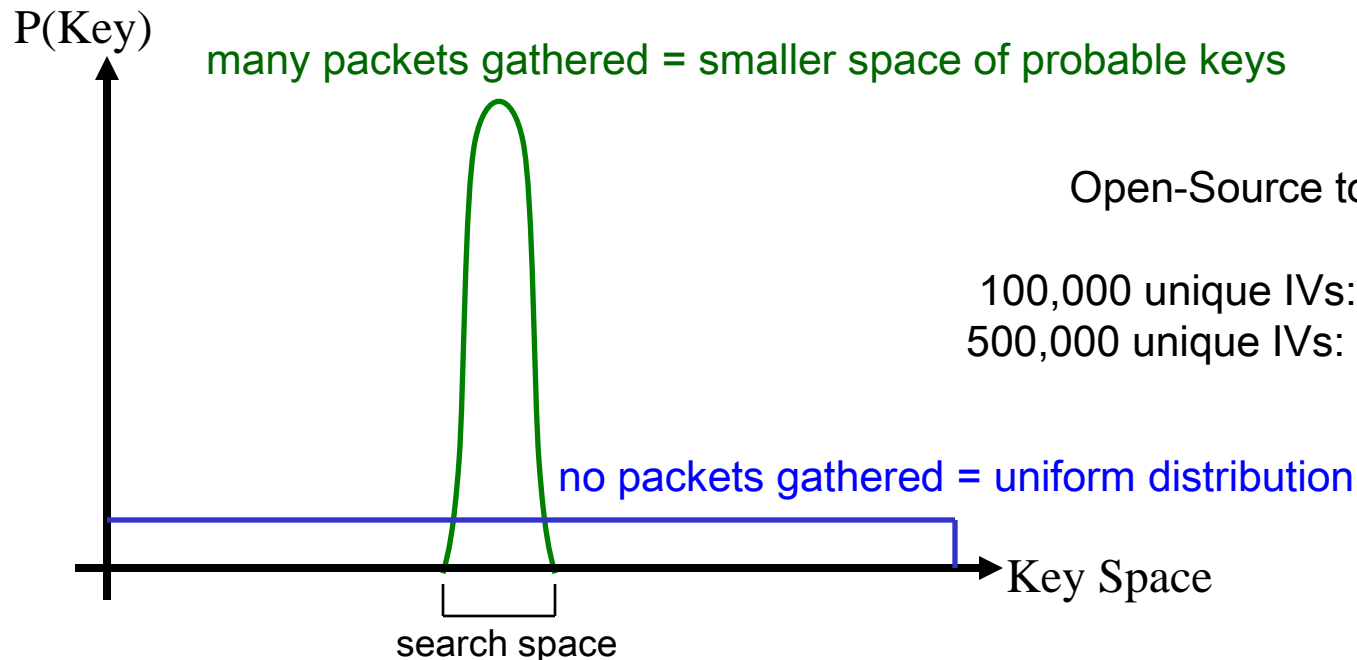
N-3 Byte Known String	4-byte CRC	Generate CRC, discard 4th byte
N Byte Encrypted String		XOR N bytes by known keystream
N Byte Encrypted String	i	Append 256 possible N+1'th bytes

If packet accepted by AP, compute  $\text{keystream}_{N+1} = i \text{ XOR } \text{CRC}_4$



# WEP: Statistical Attack

- First few bytes of keystream leak information about the key
- Collect many packets, statistically determine probable keys



Open-Source tool: **AirCrack**

100,000 unique IVs: 40-bit in < 10sec  
500,000 unique IVs: 104-bit in < 10sec



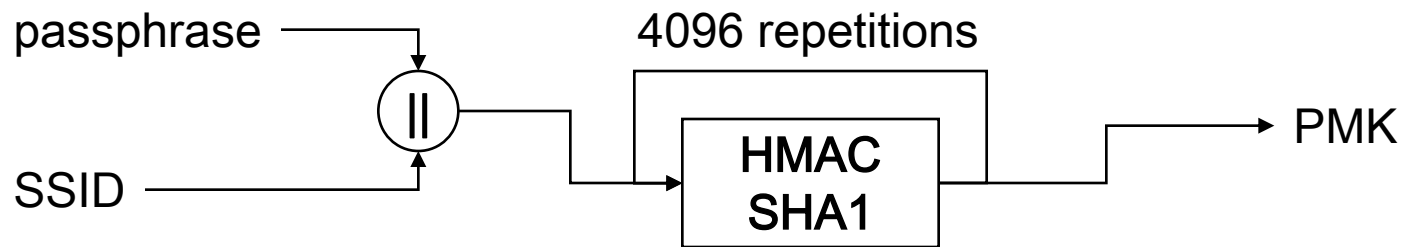
# WPA and IEEE 802.11i

- New security standards ratified in 2003, solve many of the issues
- New ciphers
  - TKIP, AES-CCM
- Mutual key derivation
- Stronger password requirements
- Generalized authentication for enterprise networks (EAP)



# Authentication: Preshared Key (PSK)

- Inputs a passphrase, outputs the pairwise master key (PMK)
- Uses PKCS5 key generation function

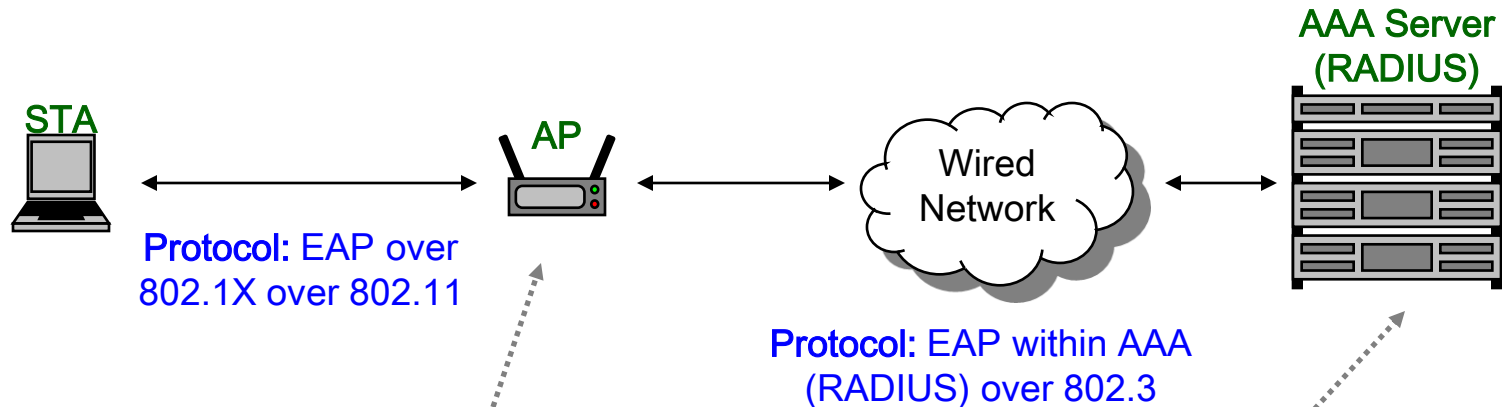


- Typical use: residential WiFi network
- Slows down dictionary attacks
  - Salting prevents precomputation



# Authentication: EAP Overview

- EAP = Extensible Authentication Protocol
- Originally developed for dialup user authentication / RADIUS



EAP Passthrough: hands all EAP data to AAA server

AAA: Authentication, Authorization, & Accounting; Validates Users, Provides Keys to AP



# Authentication: EAP Methods

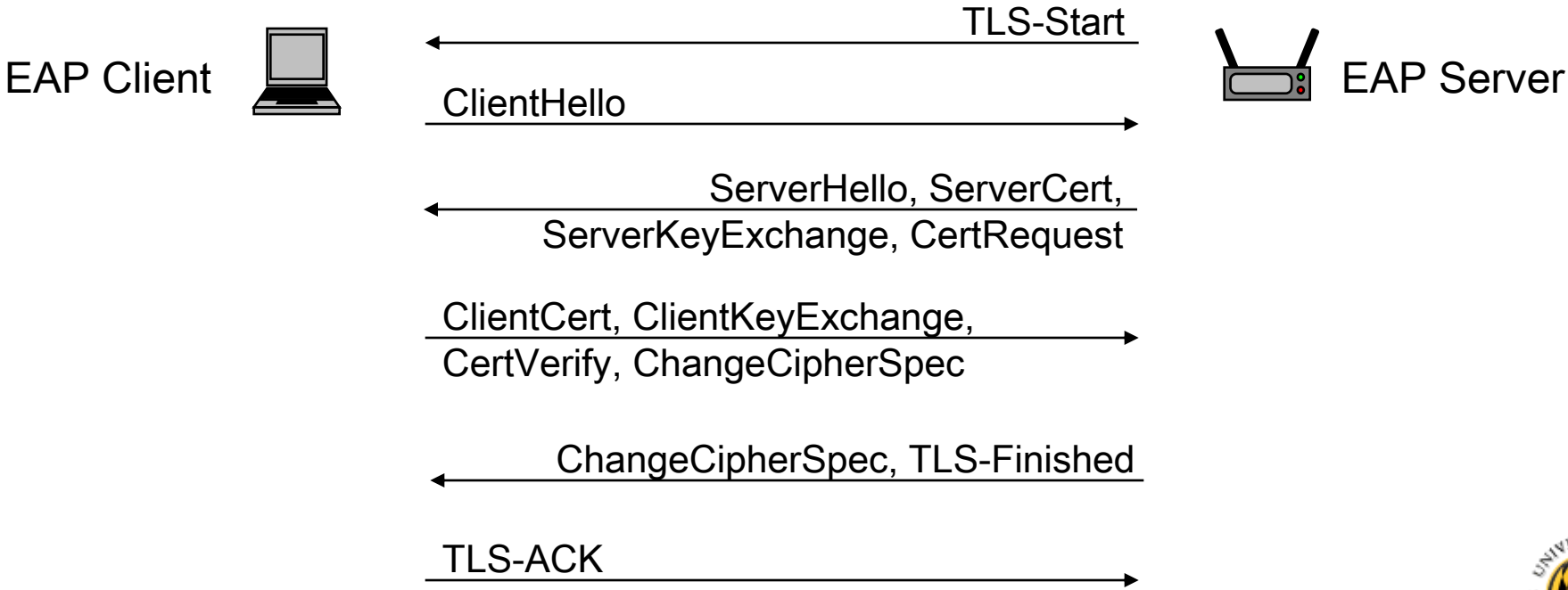
- EAP just a framework
- ***EAP Methods*** do all the real work
- Current Methods:

Method	Status	Description
TLS	RFC	PKI authentication
LEAP	Proprietary	MSCHAP auth; <b>dictionary attack</b>
PEAP	Draft	TLS tunnel, often MSCHAP auth inside
TTLS	RFC	TLS tunnel, various auths inside tunnel
FAST	RFC	basically LEAP inside TLS tunnel



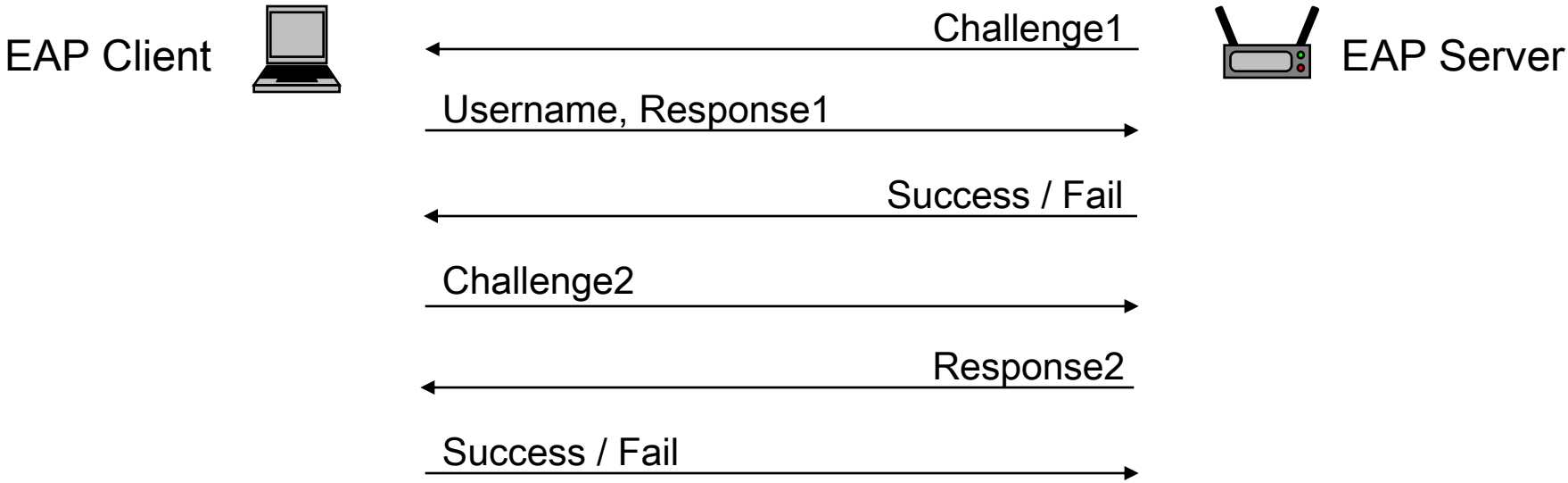
# Authentication: EAP-TLS

- Uses TLS (transaction layer security) authentication protocol and ciphersuites
- Typically public-key based, but shared-key ciphersuites emerging within IETF



# Authentication: LEAP

- LEAP = Lightweight EAP
- Cisco proprietary protocol
- Reverse engineered by Internet community, discovered it was vulnerable to a dictionary attack



# Authentication: LEAP

- It's much worse than just a basic dictionary attack -- why?
- Procedure:
  1. receive 8-byte challenge from server
  2.  $pwhash = MD4(MD4(Unicode(password)))$
  3.  $key_1 = pwhash[0..6]$
  4.  $key_2 = pwhash[7..13]$
  5.  $key_3 = pwhash[14..15] || \{0,0,0,0,0\}$
  6.  $response_1 = DES(key_1, challenge)$
  7.  $response_2 = DES(key_2, challenge)$
  8.  $response_3 = DES(key_3, challenge)$
  9. send response =  $response_1 || response_2 || response_3$



# Authentication: LEAP

- It's much worse than just a basic dictionary attack -- why?

- Procedure:

1. receive 8-byte challenge from server

2.  $\text{pwhash} = \text{MD4}(\text{MD4}(\text{Unicode}(\text{password})))$

3.  $\text{key}_1 = \text{pwhash}[0..6]$

4.  $\text{key}_2 = \text{pwhash}[7..13]$

5.  $\text{key}_3 = \text{pwhash}[14..15] \parallel \{0,0,0,0,0\}$

6.  $\text{response}_1 = \text{DES}(\text{key}_1, \text{challenge})$

7.  $\text{response}_2 = \text{DES}(\text{key}_2, \text{challenge})$

8.  $\text{response}_3 = \text{DES}(\text{key}_3, \text{challenge})$

9. send response =  $\text{response}_1 \parallel \text{response}_2 \parallel \text{response}_3$

No Salt = precompute  
dictionary of hashes

Create dictionary of password hashes  
sorted by the last two bytes

Only 16 bits of entropy =  
Brute force to easily  
recover 2 bytes of pwhash

