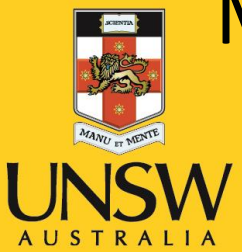


Module2, Part 2 : Symmetric Key Ciphers and WLAN Security

Securing Wireless Networks, COMP4337/9337

Never Stand Still

Professor Sanjay K. Jha

School of Computer Science and Engineering, UNSW

Today's Agenda – Part 2

- Block Ciphers
- Stream Ciphers
- How to design a flawed Security Protocol:
  - WEP Case Study
- Fixing a flawed Protocol: WPA, WPA2

Two types of symmetric ciphers

- Block ciphers
  - Break plaintext message in equal-size blocks
  - Encrypt each block as a unit
  - Used in may Internet protocols (PGP-secure email, SSL (secure TCP), IPsec (secure net-transport layer)
- Stream ciphers
  - encrypt one bit at time
  - Used in secure WLAN

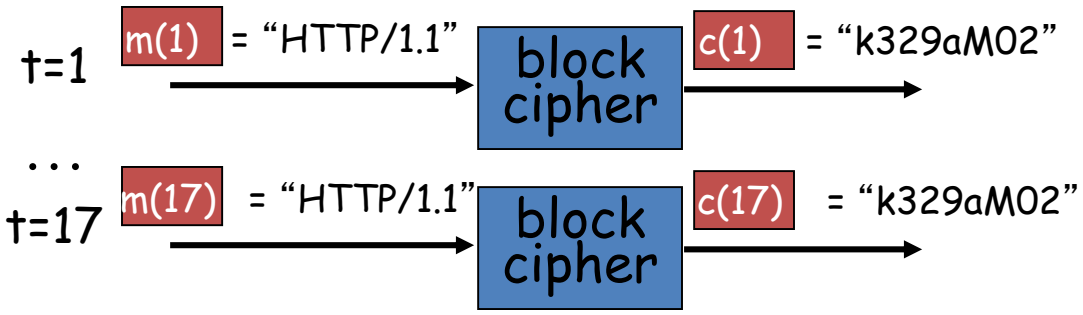
Block Cipher

- Ciphertext processed as *k* bit blocks
- 1-to-1 mapping is used to map k-bit block of plaintext to k-bit block of ciphertext
- E.g: k=3 (see table)
  - 010110001111 => 101000111001
- Possible permutations = 8! (40,320)
- To prevent brute force attacks
  - Choose large k (64, 128, etc)
- Full-block ciphers not scalable
  - E.g., for k = 64, a table with 2^{64} entries required
  - instead use function that simulates a randomly permuted table

Input	Output
000	110
111	001
001	111
010	101
011	100
100	011
101	010
110	000

Cipher Block Chaining

- cipher block: if input block repeated, will produce same cipher text:



- Sender creates a random *k*-bit number *r(i)* for *i*th block and calculates
  - $c(i) = K_S(m(i) \oplus r(i))$
  - Sends ***c(1), r(1), c(2), r(2), c(3), r(3)***….
    - *r(i)* sent in clear but *Ks* not known to attackers.
  - Example: sent text 010010010 if no CBC, sent txt = *101101101*
  - *r(001), r2(111), r3(100)*
  - Use above technique to generate cipher text *c(1) = 100, c(2) = 010, c(3) = 000*: NOTE all three output different even though same plain text 101.
  - Inefficient: twice as many bits sent

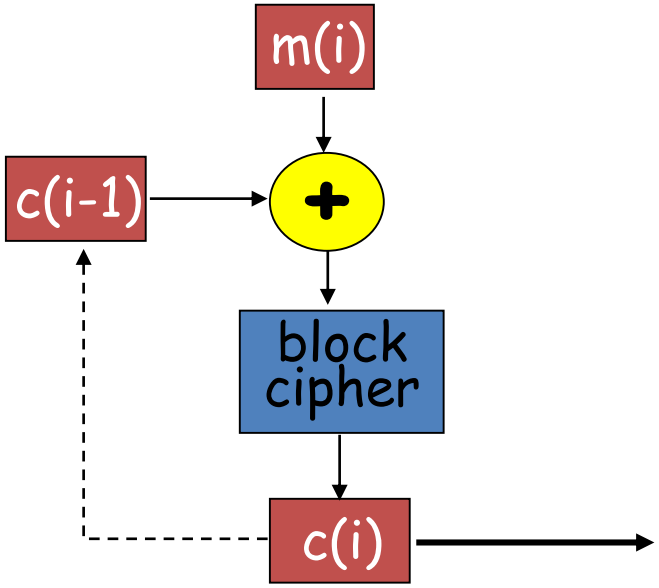
CBC: Sender

- *cipher block chaining:* XOR ith input block, $m(i)$, with previous block of cipher text, $c(i-1)$
  - $c(0)$ is an Initialisation Vector transmitted to receiver in clear
  - First block:

$$c(1) = K_S(m(i) \oplus c(0))$$

  - *Subsequent blocks:*

$$c(i) = K_S(m(i) \oplus c(i-1))$$



CBC: Receiver

- How to recover $m(i)$?
  - Decrypt with *Ks* to get $s(i) = Ks(c(i)) = m(i) \oplus c(i-1)$
  - Now the receiver knows $c(i-1)$, it can get

    $m(i) = s(i) \oplus c(i-1)$
  - **IV** sent only once
  - Intruder can't do much with **IV** since it doesn't have Ks
  - CBC has important consequence for designing secure network protocols

Block Ciphers

- Block cipher needs to wait for one block before processing it
  - Suitable for storage
- Operates on a single block of plaintext
  - 64 bits for Data Encryption Standard (DES), 128 bits for Advanced Encryption Standard (AES)
    - AES much (6x) faster than DES
- Computationally infeasible to break block cipher by brute-force by cracking the key
  - Brute force decryption (try each key)

Symmetric key crypto: DES

DES: Data Encryption Standard (US encryption standard [NIST 1993])

- 56-bit symmetric key, 64-bit plaintext input
- Block cipher with cipher block chaining
- How secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
    - DES Challenge III: Distributed. Net worked with EFF's supercomputer and a world-wide network of 100,000 PCs to crack it within 22 hours and 15 minutes, Testing 2.45 billion keys per second
  - no known good analytic attack
- Making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys

Symmetric key crypto: DES

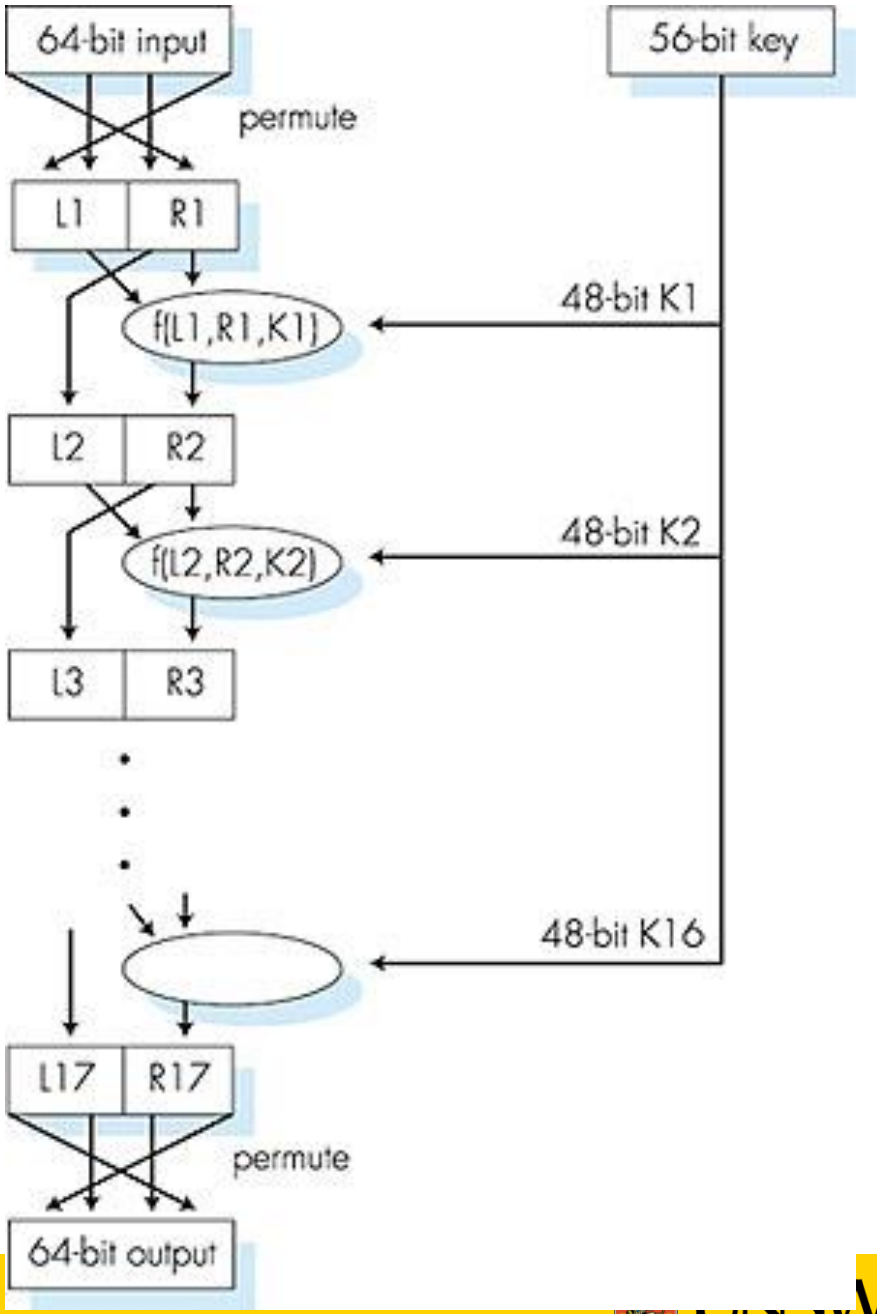
DES operation

initial permutation

16 identical "rounds" of function application, each using different 48 bits of key

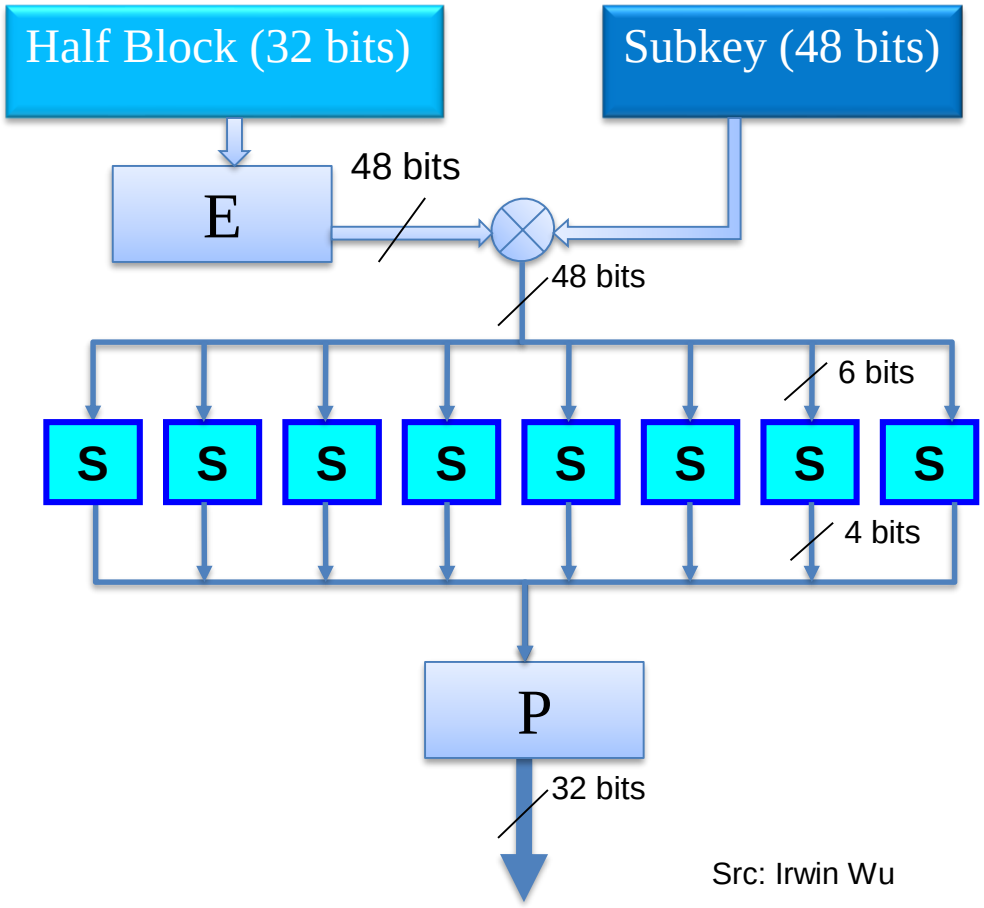
final permutation

No need to memorise this



Feistel (F) function

- Feistel (F) function
  - Expansion (E): the 32-bit half-block is expanded to 48 bits using the expansion permutation
  - Key mixing:
    - Sixteen 48-bit subkeys: one for each round from 56 bit key
    - Derived from the main key using the key schedule
    - E is combined with a subkey using an XOR operation
  - P: permutation function
    - P yields a 32-bit output from a 32-bit input by permuting the bits of the input block
  - S-box: Substitution
    - Transforms input bits using substitution tables to provide diffusion
    - Spread plaintext bits throughout ciphertext
    - Small change in either the key or the plaintext should cause a drastic change in the ciphertext (avalanche effect)



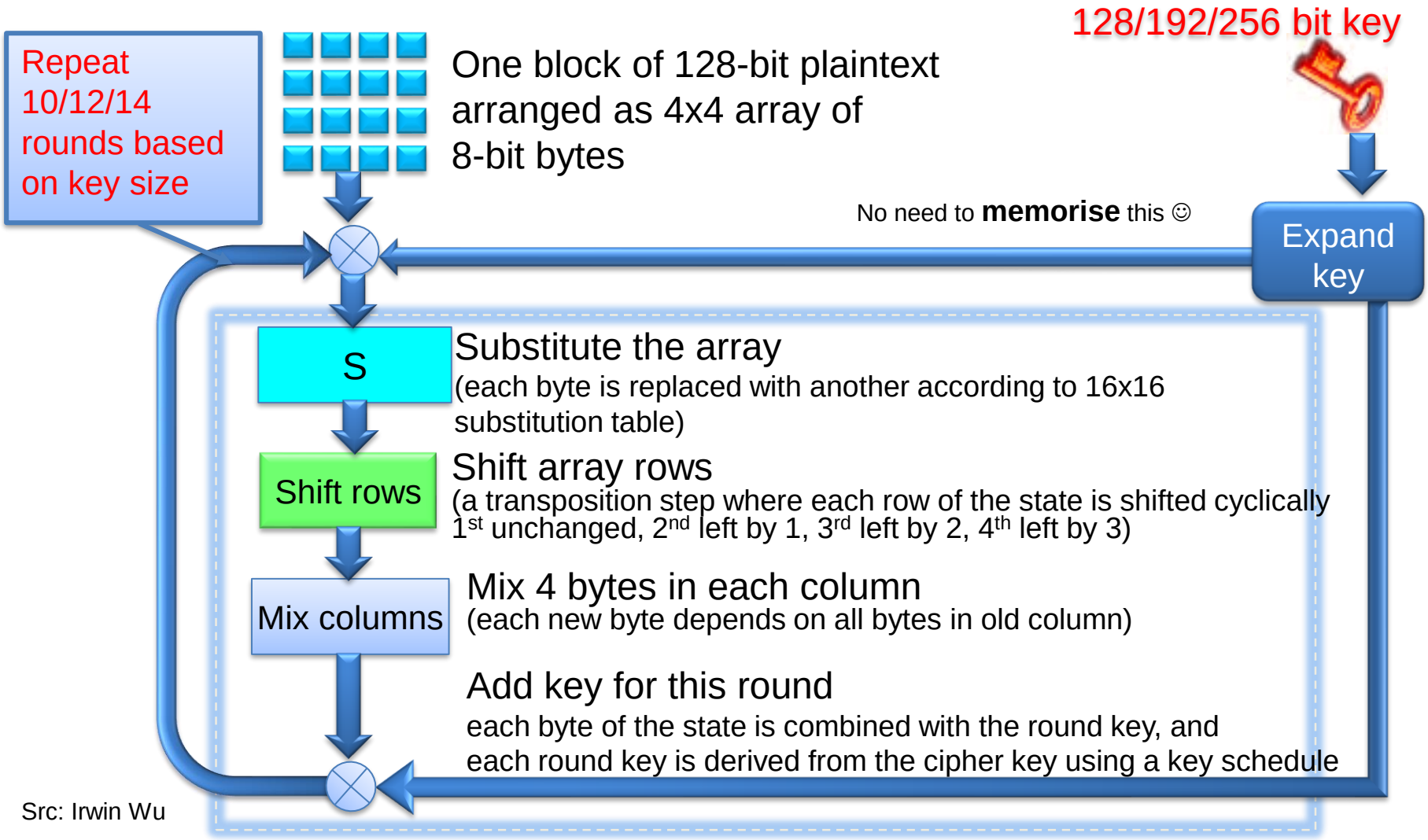
Src: Irwin Wu

No need to memorise this ☺

AES: Advanced Encryption Standard

- Symmetric-key NIST standard, replaced DES (Nov 2001)
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- Brute force decryption (try each key) taking few secs on DES, takes 149 trillion years for AES
    - Universe lifetime: 100 billion years

AES -Structure



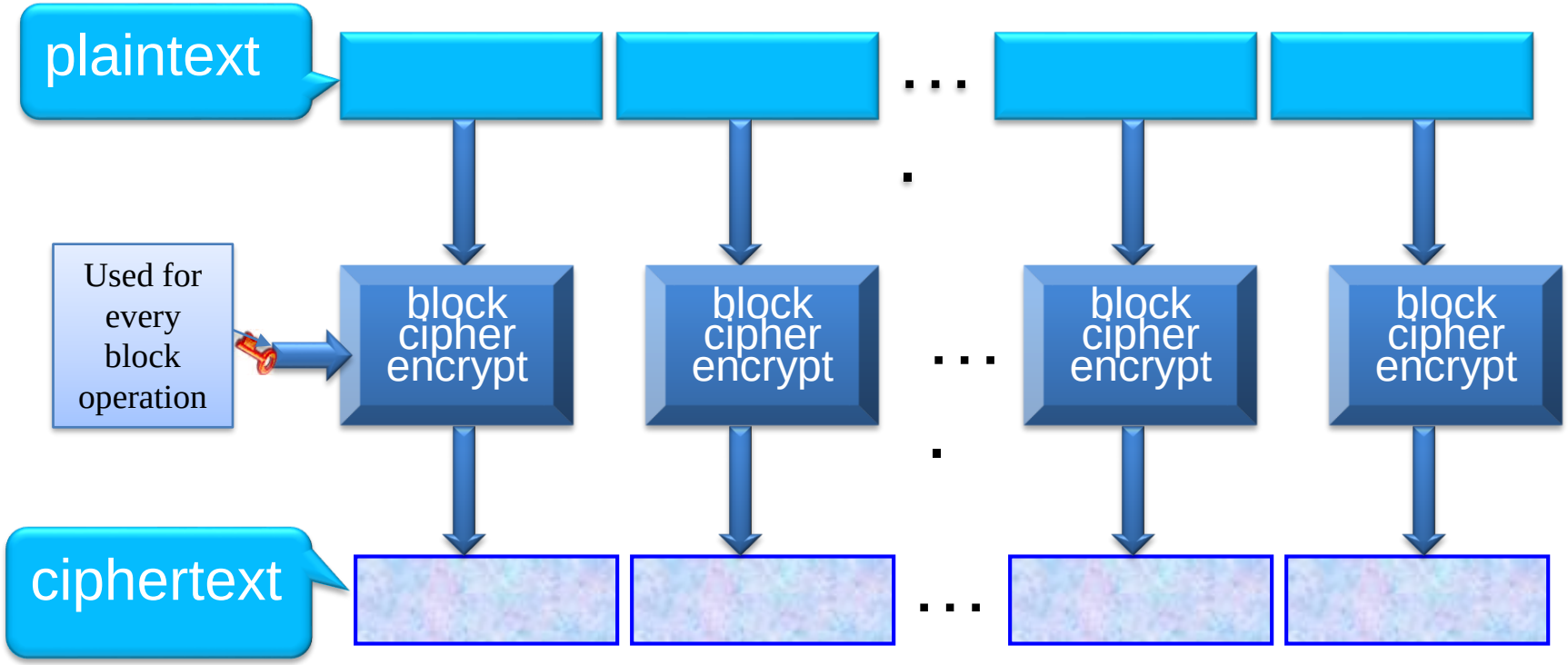
Src: Irwin Wu

AES Confidentiality Modes

- Five confidentiality modes of operation for symmetric key block cipher algorithms, such as AES
- Electronic Codebook (ECB),
  - Split plaintext into blocks, encrypt each one separately using the block cipher
- Cipher Block Chaining (CBC) mode
  - Split plaintext into blocks, XOR each block with the result of encrypting previous blocks
- Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR) modes: stream ciphers based on block cipher (will discuss as needed in future lectures)

ECB Mode Encryption

Src: Irwin Wu



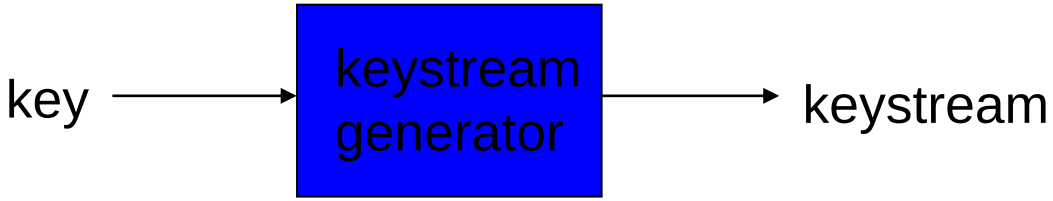
- Problem: Identical blocks of plaintext produce identical blocks of ciphertext
- No integrity checks: can mix and match blocks

Weakness of ECB

- Message repetitions may show in ciphertext
- Weakness is due to the encrypted message block uniquely mapped to the plaintext block
- Main use is sending only a few blocks of short data

- *If longer messages, use Cipher Block Chaining (as discussed earlier)*

Stream Ciphers



- Process message bit by bit (as a stream)
  - Ideal for real-time communication
  - A keystream must not be reused; otherwise the encrypted messages can be recovered
- *combine each byte of keystream with byte of plaintext to get ciphertext:*
  - m(i) = ith unit of message
  - ks(i) = ith unit of keystream
  - c(i) = ith unit of ciphertext
  - $c(i) = ks(i) \oplus m(i)$ ($\oplus$ = exclusive or)
  - $m(i) = ks(i) \oplus c(i)$

Rivest Cipher 4 (RC4)

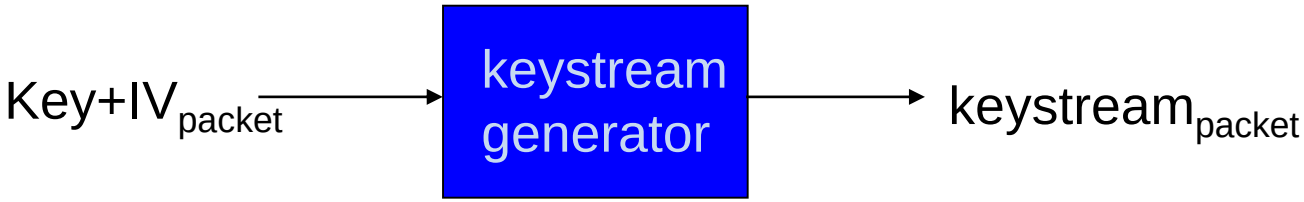
- Rivest Cipher 4: Designed by Ron Rivest
  - A proprietary cipher owned by RSA.com
  - No longer a trade secret
  - Ideal for software implementation, as it requires only byte manipulations
- Variable key size (40 to 256 bits), byte-oriented stream cipher
- Widely used
  - SSL/TLS, Wireless WEP and WPA, Cellular Digital Packet Data, OpenBSD pseudo-random number generator

Wired Equivalent Privacy (WEP)

- Provide security equivalent to Wired Network
  - *Problem starts with this thinking!*
- Symmetric key crypto
  - confidentiality
  - end host authorization
  - data integrity
- Efficient
  - implementable in hardware or software

Stream cipher and packet independence

- Design goal: each packet separately encrypted
- If for frame $n+1$, use keystream from where we left off for frame n, then each frame is not separately encrypted
  - need to know where we left off for packet n (e.g Cipher Block chain approach)
- WEP approach: initialize keystream with key + new IV for each packet:

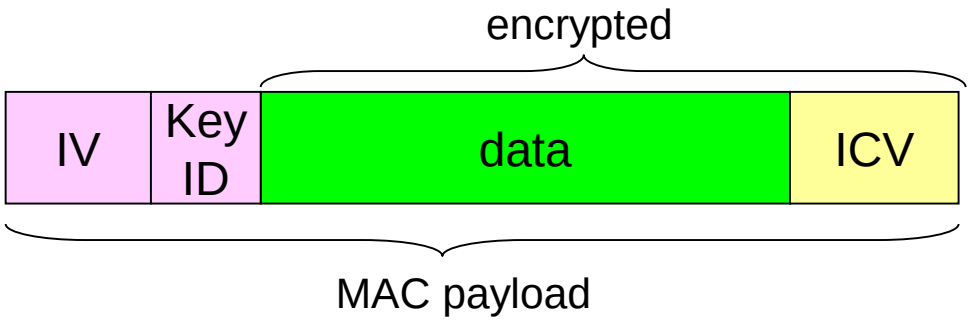


WEP Pre-shared Key

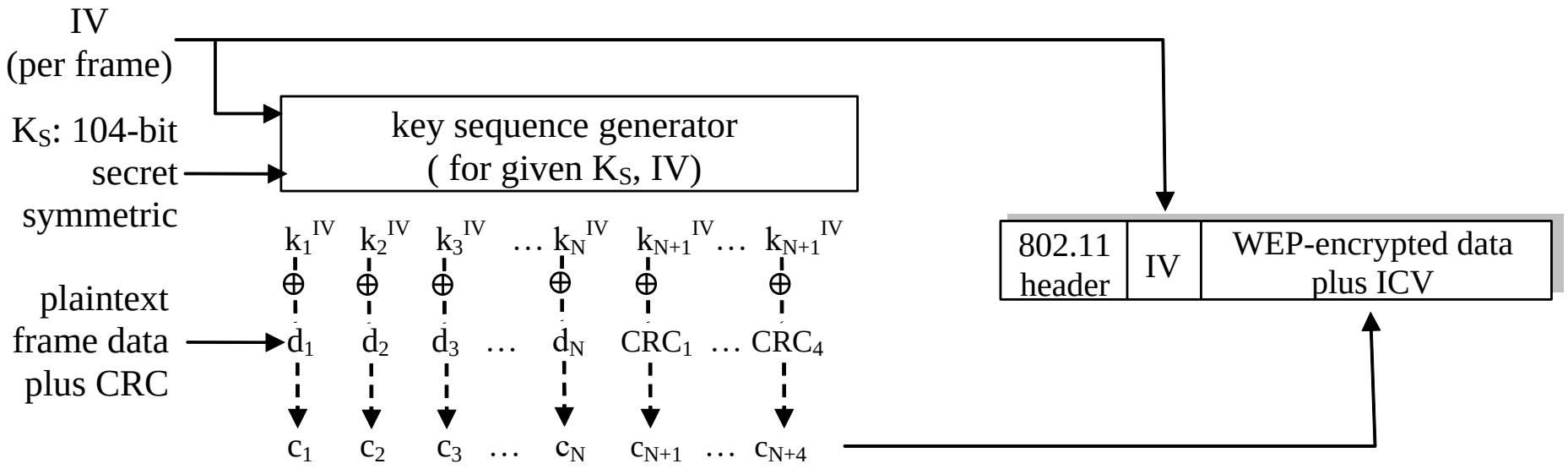
- Enter a key (password) on access point and then then enter the key on all devices
  - This is the pre-shared key, AKA WEP Key *(Shared Secret)*.
- Not possible to authenticate individuals
  - hard to distinguish who is using service - needs extra work.
- A key compromise for one user means that every device needs to change new key
  - Must be distributed to all users securely

WEP encryption (1)

- sender calculates Integrity Check Value (ICV) over data
  - four-bytes for data integrity: uses CRC-32
- each side has 104-bit shared key
- sender creates 24-bit initialization vector (IV), appends to key: gives 128-bit key
- sender also appends keyID (in 8-bit field)
- 128-bit key input into pseudo random number generator PRNG e.g. RC4 to get keystream
- data in frame + ICV is encrypted with RC4:
  - Bytes of keystream are XORed with bytes of data & ICV
  - IV & keyID are appended to encrypted data to create payload

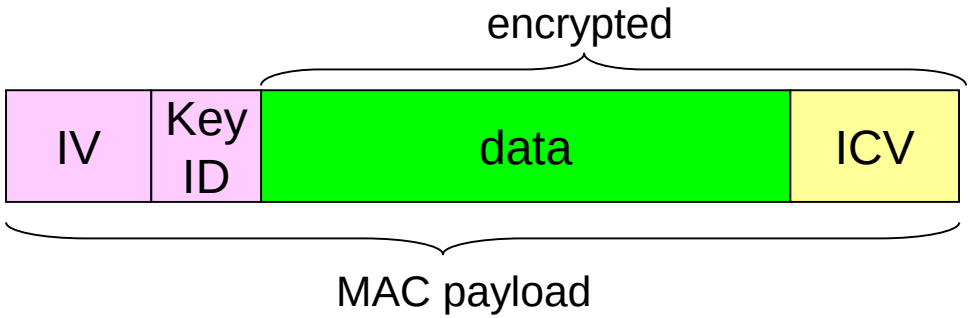


WEP encryption (2)



new IV for each frame

WEP decryption

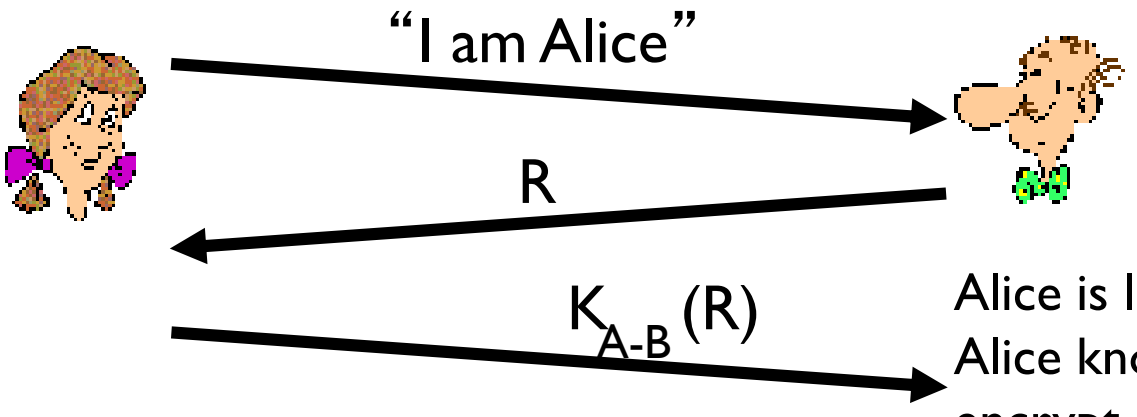


- receiver extracts IV (*received in plaintext*)
- inputs IV, shared secret key into pseudo random generator, gets keystream
- XORs keystream with encrypted data to decrypt data + ICV
- verifies integrity of data with ICV
  - note: message integrity approach used here is CRC-32 different from MAC (message authentication code) and signatures (using PKI).

End-point authentication w/ nonce

Nonce: number (R) used only *once –in-a-lifetime*

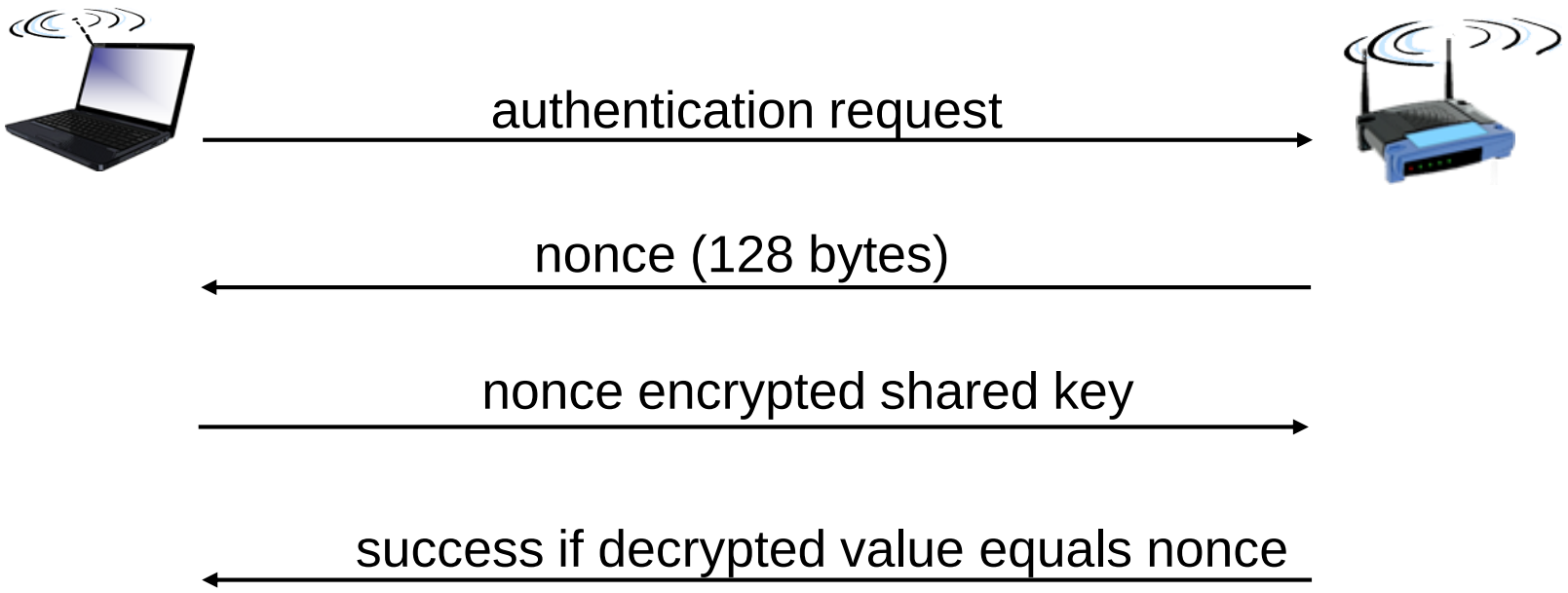
How to prove Alice "live": Bob sends Alice *nonce*, R. Alice must return R, encrypted with shared secret key



Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

Nonce: *Handy tool for secure protocol design*

WEP authentication



Notes:

- not all APs do it, even if WEP is being used
- AP indicates if authentication is necessary in beacon frame
- done before association

Breaking 802.11 WEP encryption

Security hole:

- 24-bit IV, one IV per frame, -> IV's eventually reused
  ~16Million IVs at high speed exhausted in 2 hours
- IV transmitted in ***plaintext*** -> IV reuse detected

Attack:

- Trudy causes Alice to encrypt known plaintext $d_1\ d_2\ d_3\ d_4\ \ldots$
- Trudy sees: $c_i = d_i$ XOR k_i^{IV}
- Trudy knows $c_i\ d_i$, so can compute $k_i^{IV} = d_i$ XOR c_i
- Trudy knows encrypting key sequence $k_1^{IV}\ k_2^{IV}\ k_3^{IV}\ \ldots$
- Next time IV is used, Trudy can decrypt!

Problem with Linear Checksum

- Encrypted CRC-32 used as integrity check Vector (ICV)
  - Fine for random errors, but not malicious ones
  - Bits can be changed in packet without decrypting

- An attacker can change encrypted content (substitute by gibberish), compute a CRC over the substituted text and produce an 802.11 frame that will be accepted by the receiver.

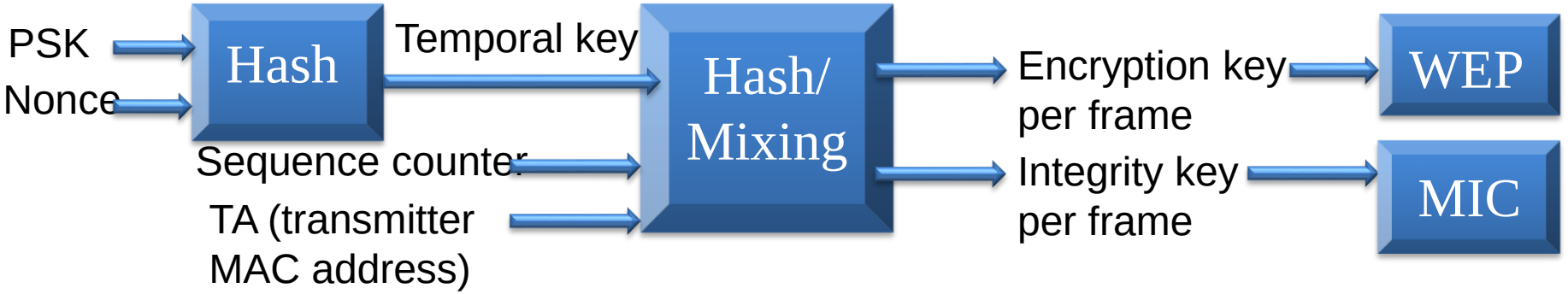
Wi-Fi Protected Access (WPA)

- Two versions WPA and WPA2
  - WPA temporary solution to fix WEP while WPA2 developed
- WPA compatible with existing hardware that supported WEP
- WPA uses Temporal Key Integrity Protocol (TKIP)
  - Used RC4 for compatibility
  - Every packet encrypted with unique encryption key

802.11i: WPA – New Features

- To provide stronger authentication than in WEP:
  - Special purpose Message Integrity Code (MIC) as opposed to WEP CRC
- To prevent Fluhrer, Mantin and Shamir (FMS) aka FMS-style attacks
  - a new *per-frame key is* constructed using a cryptographic hash
- Temporal Key Integrity Protocol (TKIP) uses a cryptographic mixing function to combine a temporal key, the TA (transmitter MAC address), and the sequence counter into the WEP seed (128 bits)
  - Pre Shared Key (PSK) AKA WPA-Personal similar to WEP-Key
    - However, it is not used for encryption
  - Instead, PSK serves as the seed for hashing the per-frame key

802.11i: WPA Contd.



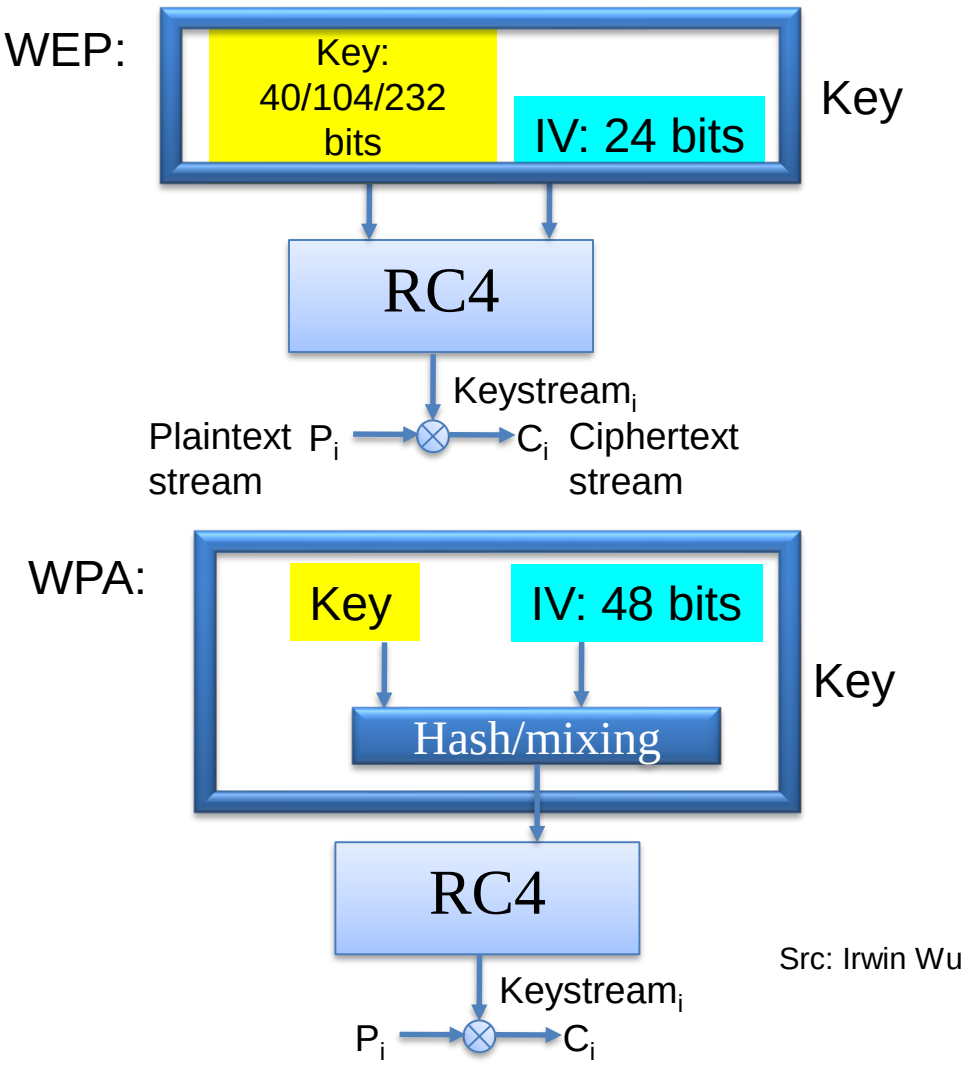
Src: Irwin Wu

- TKIP changes the per packet key completely after every single packet
  - One key for encryption (128 bits)
  - One key for integrity (64 bits)
- The WEP IV is extended to 48 bits, and used as a packet sequence counter
  - A per packet sequence counter is used to prevent replay attacks
  - If a packet is received out of order, it is dropped by the receiving station

802.11 frame with WPA

Recap: WEP vs WPA security

- WPA temporary solution to fix WEP while WPA2 developed
- WEP IV extended to 48-bit IV
  - Reuse > 100 years for replay of the same IV
- RC4 key = Function(WPA Key||IV)
  - Every packet encrypted with unique encryption key
- IV used as a packet sequence space to prevent replay attack



Src: Irwin Wu

802.11i: WPA2

- WPA2 2004
  - New AP hardware, 30 Million Instructions/sec, RC4 off-load hardware doesn't do AES or CCMP
  - AES-CCMP 128-bit AES
    - CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol)
  - Improved 4-way handshake and temporary key generation
- We may cover some details of WPA2 but for now, if you have WPA2, this would be the safest option to use.
- WPA- Enterprise network security (802.1X) in later weeks

Acknowledgements

- *Acknowledgement: foils are adapted mainly from* **Introduction to Computer Networks and Cybersecurity by Wu and Irwin, CRC Press (Chapter 21)**

- *Some foils are also from Günter Schäfer,* **Security in Fixed and Wireless Networks**, *Wiley (new edition available in German only, English in 2015)*
- *A few foils are from Adrian Perrig (ETH)*

WPA PSK Weakness (self-read)

- WPA, using the Temporal Key Integrity Protocol, was cracked by Erik Tews and Martin Beck
  - More in lab
- Thomas Roth demonstrated at the 2011 Black Hat conference that WPA PSKs can be cracked quickly and easily using Amazon's Elastic Compute Cloud (EC2) service
  - He cracked his neighbor's WPA password in 20 minutes using a dictionary attack and a list of 70 million words
  - The attack only required one instance of Roth's self-made Cloud Cracking Suite (CCS) tool running in the cloud
  - It reached about 50,000 PSKs/s
- The EC2 uses 400 cloud CPUs to launch a dictionary attack on a WPA key for $17
  - http://www.wpacracker.com/
  - The attack is based on a list containing 135 million entries which can be extended to include such optional extras as a German dictionary or an extended English language word list (284 million entries)

Message Forgery (Self-Read)

- CRC-32 is *linear*, which means that it is possible to compute the bit difference of two CRCs based on the bit difference of the messages over which they are taken.

- Flipping bit n in the message results in a deterministic set of bits in the CRC that must be flipped to produce a correct checksum on the modified message.

- Because flipping bits carries through after an RC4 decryption, this allows the attacker to flip arbitrary bits in an encrypted message and correctly adjust the checksum so that the resulting message appears valid.
- Implications:
  - "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

FMS Attack (self-read)

- Fluhrer, Mantin and Shamir (FMS) attack
  - For 50% success rate, capture around 5 Million packets on average
  - Due to inherent weakness in RC4, output of encrypting with first few bytes of key not random
  - Certain key values generate predictable pattern of encrypted data
    - Associated packets are IVs are “weak”
  - Initial determine first bytes of key through IVs and then get the rest through statistical analysis
  - Encrypted ARP packets can be captured and replayed to get encrypted ARP response
    - more in lab

Hotspot Security (self read)

- For most hotspots: Unfortunately almost none!
- If you do not have to configure any security parameters besides typing in a username and password in a web page, expect the following:
  - The hotspot operator checks your authenticity at logon time (often protected with SSL to protect against eavesdropping on your password)
  - Only authenticated clients will receive service as packet filtering is deployed to only allow accessing the logon page until successful authentication
  - Once logon authentication has been checked: no further security measures
  - No protection for your user data:
- Everything can be intercepted and manipulated
  - However, you can deploy your own measures, e.g. VPN or SSL, but configuration is often tedious or not even supported by communication partner and performance is affected because of additional (per-packet-) overhead
    - Plus: your session can be stolen by using your MAC & IP addresses!