

CMPUT 299 (B1) Winter 2008

# **Security in a Networked World**

*WEP: A Case Study*

yannis@cs.ualberta.ca

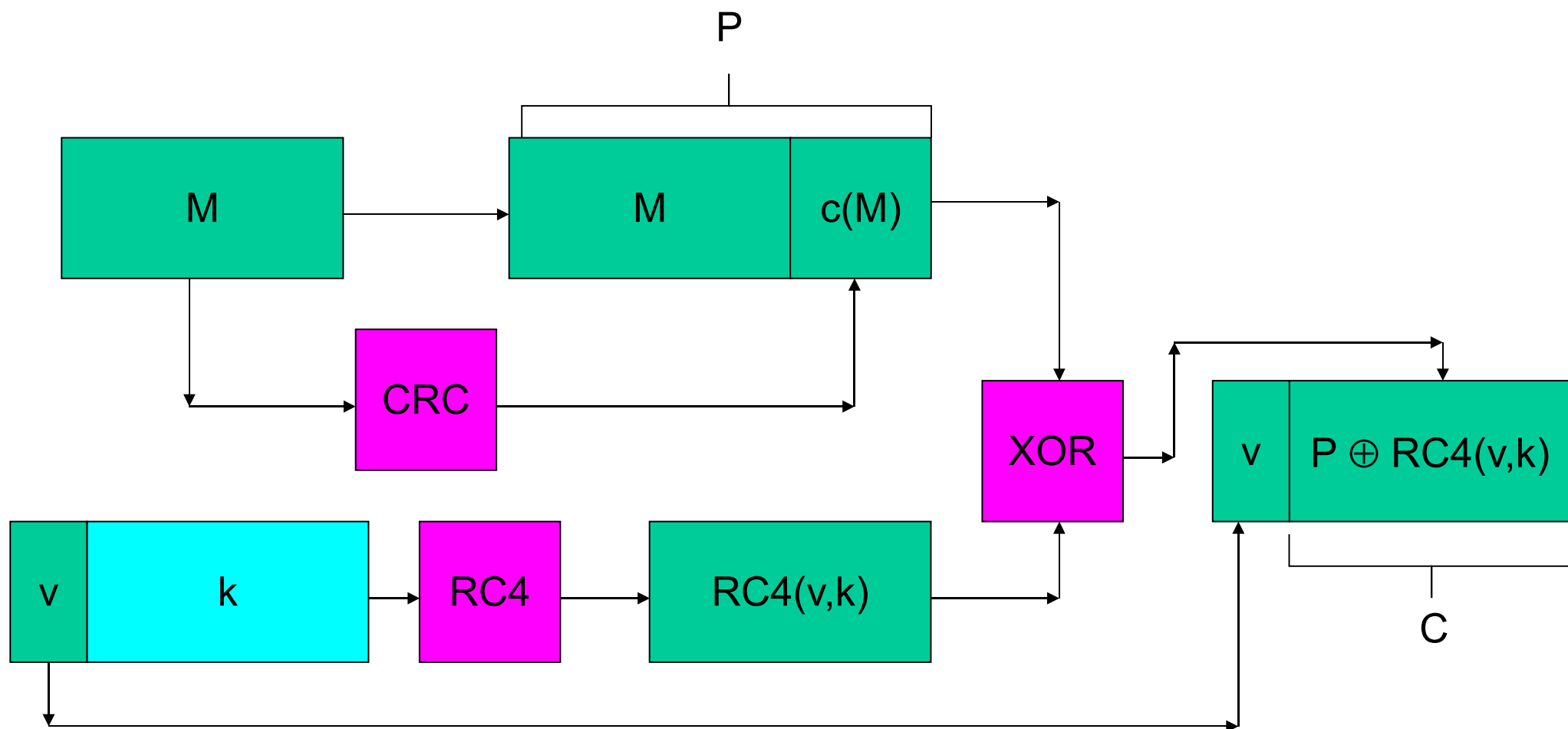
# Wired Equivalent Privacy (WEP)

- Confidentiality
  - Access control
  - Integrity
  - Non-repudiation
  - Authenticity
  - Availability
- } WEP's objectives

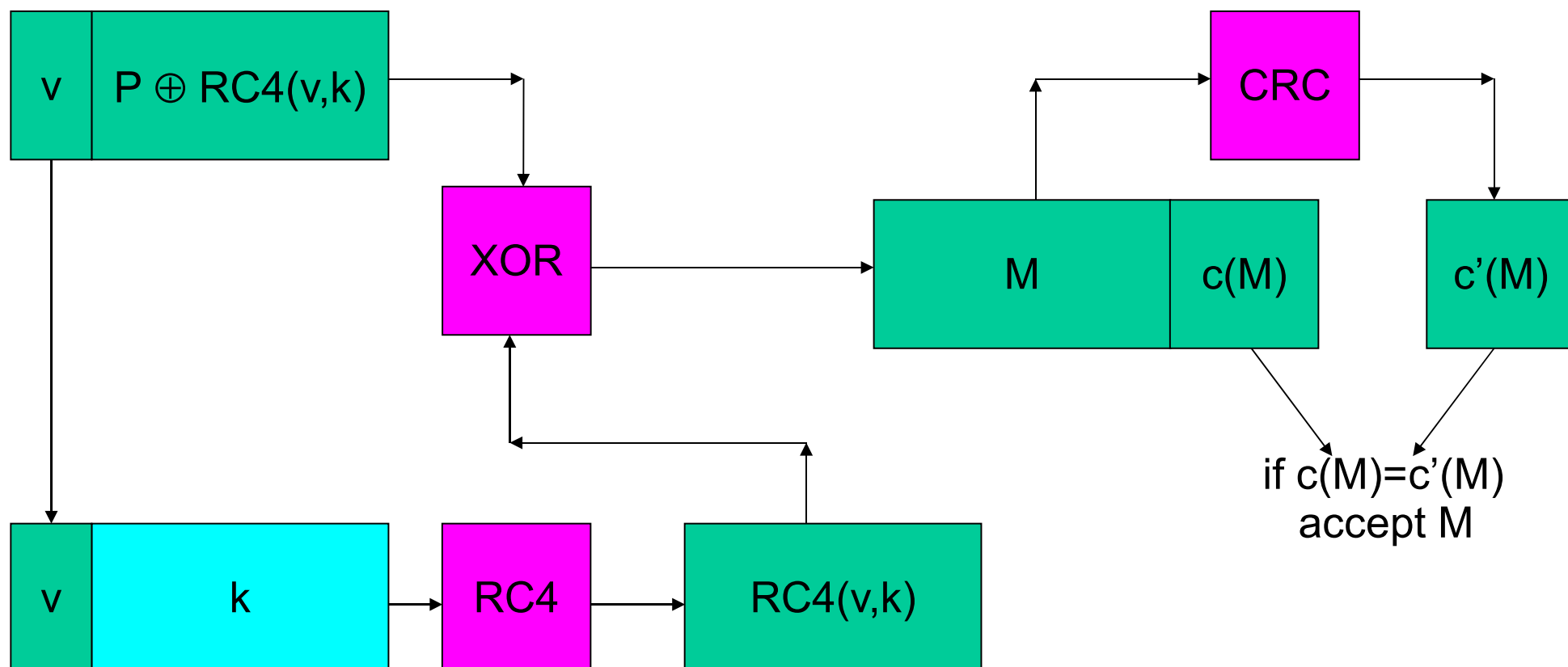
# The Source of the Problem

- ▶ The need for fast hardware implementation along with minimum overheads (including coordination and control message exchanges) results in compromises (cutting corners)
- ▶ The encryption must be in real-time, easy to implement in hardware, with little computation and energy expenditure involved.

# WEP Block Diagram (Sender)



# WEP Block Diagram (Receiver)



## WEP (some notation)

- ▶ Upon receipt:

- ▶ generate  $RC4(v,k)$

- ▶  $P' = C \oplus RC4(v,k)$

- ▶ If  $C$  was correctly received then:

$$P' = C \oplus RC4(v,k) = (P \oplus RC4(v,k)) \oplus RC4(v,k) = P$$

- ▶ Break  $P'$  into  $M$  and  $c$

- ▶ Check if  $c = c(M)$

- ▶ If yes, accept  $M$  as the message that was sent.

# The Two-Time-Pad Vulnerability

- Golden rule of stream ciphers: you must *never* encrypt two messages using the same keystream  $K$ . (If you end up reusing keystreams, make every effort to not indicate that this has happened.)
- If  $C1 = P1 \oplus K$ , and  $C2 = P2 \oplus K$
- Then  $C1 \oplus C2 = P1 \oplus K \oplus P2 \oplus K = P1 \oplus P2$
- Cryptanalysis of XOR-ed plaintext is easy.

## The Two-Time-Pad Vulnerability (cont'd)

- First, WEP announces **openly** its reuse of the IV (!) and we can bet that the shared key rarely changes. An adversary can tell with high confidence when the  $v$  and  $k$  are the same, and hence when the  $RC4(v,k)$  is the same.
- More to the point the **IV sequence is rather short** ( $2^{24}$ ) so you are almost certainly going to reuse IV values. Worse, hardware picks the successive IV values in a naïve way (possibly sequentially) and typically may not preserve “state” since last power-up.
  - Sequential IV: you wrap around after approx 16 million packets.
  - Random IV: you are bound to find a “collision” after ~4000 packets. (Birthday Paradox)
  - Many clients: a (possibly large) set of nodes use the same key!

# The Adversary's Strategy (1)

- Knowing C and P (as well as the corresponding v) could allow you to reproduce the keystream without knowing the key!  $RC4(k,v) = P \oplus C$
- Controlling P and matching it to C is easier than it appears with some control over higher layer protocols. For example, you can send ping packets, or spam, or anything innocuous.
- It can also be performed passively by watching for collisions and performing cryptanalysis to infer P.
- Note: no need to know the value of the shared secret k.
- The results of the above process can be used to populate a table that provides the  $v \rightarrow RC(v,k)$  mapping.

# Adding Insult to Injury: the CRC

- A correctly decrypted packet contains a correct checksum (making the task of identifying that we correctly decrypted a packet even easier!).
- CRCs are linear functions  $c(M \oplus D) = c(M) \oplus c(D)$ . Also, the CRC calculation is independent of any shared (key) secret. This makes the CRC unsuitable for authentication purposes (e.g. in contrast to MACs). Our only hope is to use CRC as a means to ensure integrity.
- However, the nature of random errors is different from that of intentional errors. The intentional (integrity intrusion) errors are introduced in a systematic manner. Assume we would like to flip a bit, and  $D$  corresponds to the intended change. We can calculate  $c(D)$  as well because the CRC polynomial is known (defined in the standard).

## The Adversary's Strategy (2)

- Modify the contents of a packet that will pass as a valid one even if you do not know the keystream used! (i.e., we break the integrity guarantees)
- Capture the ciphertext  $C$  of a message  $M$ 
  - Remember  $C = \text{RC4}(v,k) \oplus \langle M, c(M) \rangle$
- The adversary will construct a  $C'$  that decrypts to an  $M'$  which is accepted by the receiver and
  - It will be  $M' = M \oplus D$  with  $D$  selected by the adversary

# Message Modification

- It is enough to perform  $C' = C \oplus \langle D, c(D) \rangle$ 
  - =  $RC4(v, k) \oplus \langle M, c(M) \rangle \oplus \langle D, c(D) \rangle$
  - =  $RC4(v, k) \oplus \langle M \oplus D, c(M) \oplus c(D) \rangle$
  - =  $RC4(v, k) \oplus \langle M', c(M \oplus D) \rangle$
  - =  $RC4(v, k) \oplus \langle M', c(M') \rangle$
- $M'$  is accepted as being valid.

## Attacking Integrity

- Message injection is also easy. One compromised plaintext to ciphertext mapping (for a corresponding  $v$ ) is enough. We can use the same  $v$  as that of the compromised message.
  - Remember  $P \oplus C = P \oplus RC4(v,k) \oplus P = RC4(v,k)$
- We can construct  $M'$  and  $P' = \langle M', c(M') \rangle$
- Then apply the keystream  $C' = RC4(v,k) \oplus P'$
- And ship it to the destination.
- Note that the standard calls for the IV values to be respected – regardless of the fact that they are reused.

## The (Weak) Authentication Protocol

- The objective is for an AP to verify that a node is in possession of the secret key without revealing the key to eavesdroppers. It is a classic application of a *challenge-response* protocol.
  - The AP sends a cleartext string to the client  
B → A: M
    - Can be a random sequence of bits.
    -
  - The client sends the result of encrypting the challenge with the secret key to “convince” the challenger that it knows the key  
A → B: v,  $\langle M, c(M) \rangle \oplus \text{RC4}(v, k)$

## The Adversary's Strategy (3)

- An adversary can record the challenge/response exchange for a given unknown key  $k$ ; and can therefore extract  $v$  and  $RC4(v,k)$ .
- The adversary can now authenticate itself with the AP. The base station will send a challenge string  $M'$  to the adversary.
- The adversary replies with
  - $v, \langle M', c(M') \rangle \oplus RC4(v,k)$
- The AP will have to accept the adversary.

# Weak Confidentiality

- Since (en/de)ryption are symmetric, an adversary can decrypt packets with the aid of the AP. Note that we can authenticate with the AP so we are legitimate members if we so wish!
- Two techniques are possible:
  - IP Redirection
  - Reaction attacks

# IP Redirection

- Modify a captured packet to include your own IP address as its destination. Transmit the changed packet to the AP. The AP will decrypt the packet and route it to the IP address that is under the attacker's control.
- We know how to modify the packet without violating the CRC integrity but we do not know what it was in the first place. The fields of interest are
  - (a) the destination address
    - Easy to determine if interested in incoming (downlink traffic)
  - (b) the checksum of the IP header
    - If original IP address = DH, DL
    - And the modified IP address = DH', DL'
    - Then  $d' = d + DH' + DL' - DH - DL$  (one's complement)
    - (Notation:  $\Delta = d \oplus d'$ )

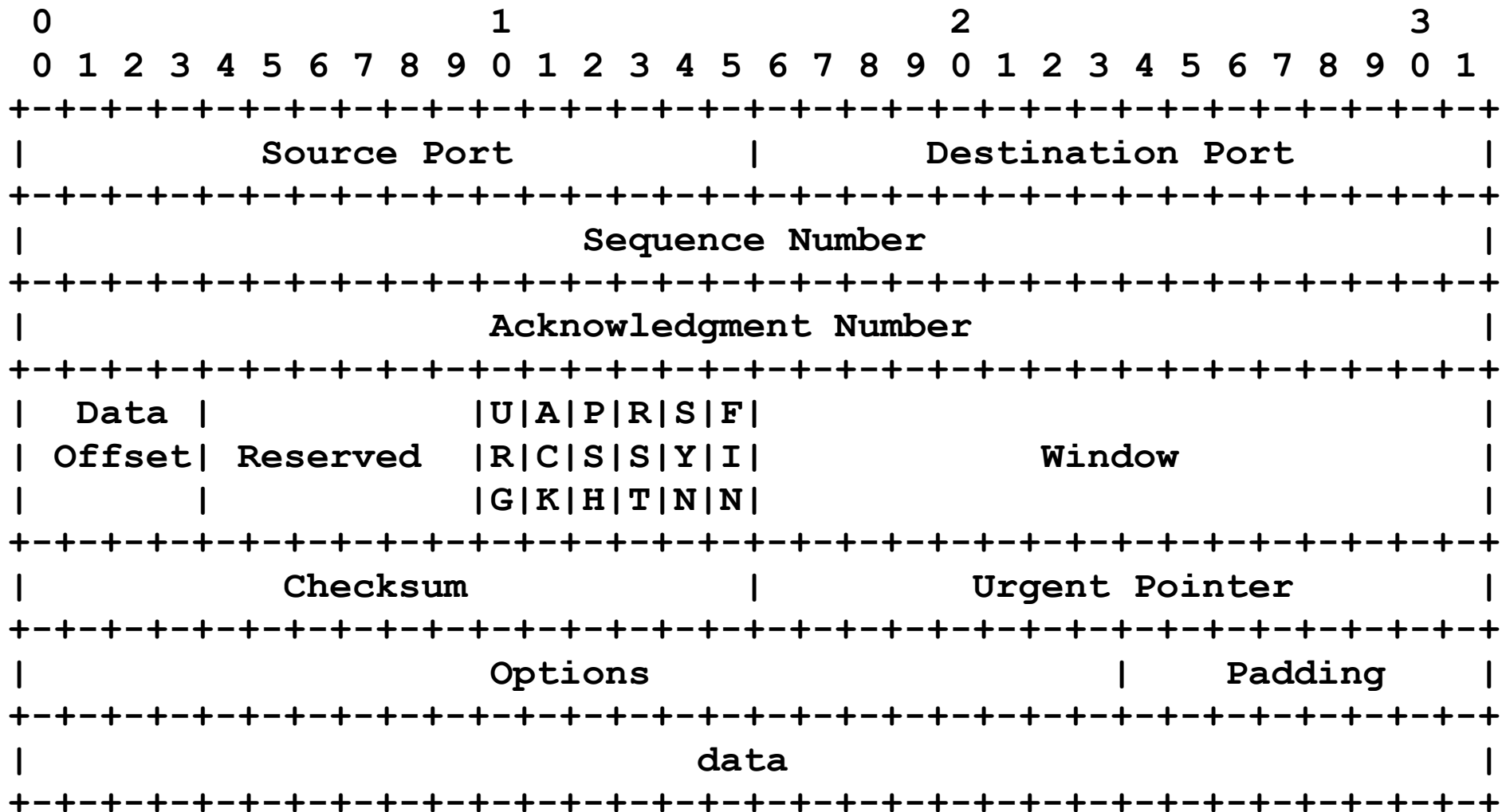


# IP Redirection (cont.)

- Three possibilities for dealing with  $d$  to  $d'$  transformation.
  - Maybe the IP checksum  $d$  is known, then we know which bits to flip to get  $d'$
  - If not known only a subset of  $\Delta$  values are common (you can explore the possibilities off-line).
  - Make it so  $\Delta=0$  (identical IP checksums) by modifying other fields.

# Reaction Attacks

- We need to consider only messages that are TCP packets. A TCP packet with invalid TCP checksum is silently dropped. Else, we receive in response an ACK packet (easy to identify by its short size).
- The strategy is once we intercept  $\langle v, C \rangle$  and flip some bits of  $C$  and recompute checksum, send it to the AP and wait to see if ACK is sent back.
- It turns out, the bits to flip to ensure the *TCP checksum* remains undisturbed are
$$P_i \oplus P_{i+16} = 1$$
- The natural conclusion is to use this strategy to flip bits (on a trial and error basis) that reveal, one by one, the original plaintext bits.



# Reaction Attacks (cont.)

- TCP CRC = 1's complement addition of the 16-bit words of message  $M$ .
- 1's complement addition  $\sim$  modulo  $2^{16} - 1$
- $C' = C \oplus D$ ,  $D$  = bit positions to flip.
- We choose  $D$ : pick  $i$  arbitrarily, set positions  $i$  and  $i+16$  of  $D$  to 1 and the rest to 0.
- Convenient property:  $P \oplus D = P \text{ mod } 2^{16}-1$  holds when  $P_i \oplus P_{i+16} = 1$

# Solutions (?)

- Longer IVs
- IVs that are never repeated while a key is in use.
- IVs not duplicated across machines using the same key.
- Frequent changes of the shared key.
- The frame checksum should be based on a Message Authentication Code (instead of the CRC) which depends on the secret key and the IV.

## A “synchronous” attack

- Start with an easily guessable higher layer communication between AP and mobile.
  - DHCP, since it is most often the first to be exchanged between node and AP. Both DHCP requests and replies have predictable contents. DHCP requests more so.
    - Even apply a “gross” filtering criterion. Like waiting for a frame with length 802.11 header + LLC (data link) header info + IP header + UDP header + DHCP request length.
    - More filtering: recipient in 802.11 header is the broadcast address ff:ff:ff:ff:ff:ff and the packet contains the ToDS and WEP flags set in the 802.11 header. Also the sender’s MAC address recently appeared in the network.

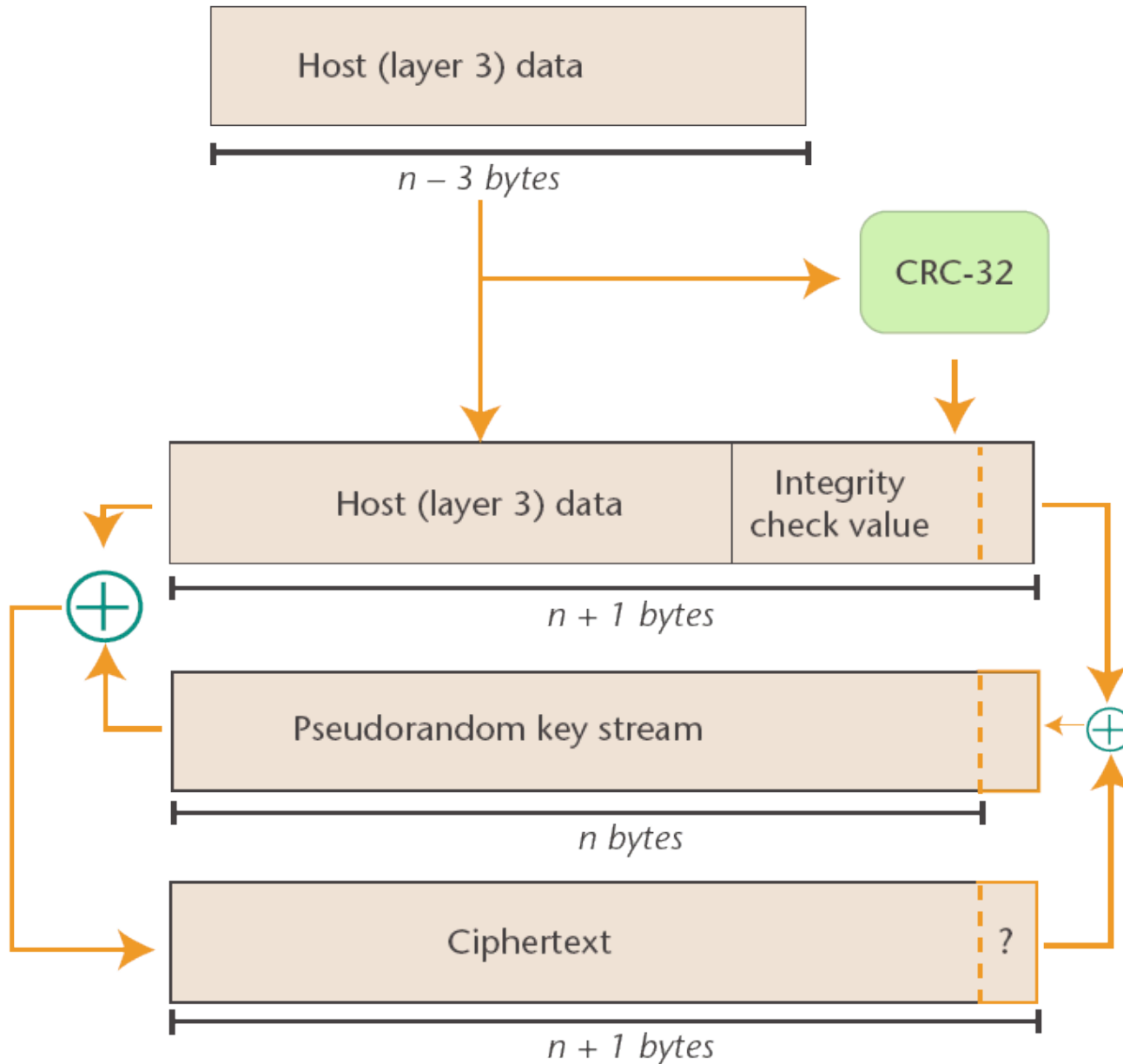
# The predictable DHCP request

| FIELD                  | BYTES | COMMENT                                   |
|------------------------|-------|---|
| Logical link control   | 8     | Same for all IP packets                   |
| IP Header              |       |   |
| Version/header length  | 1     | Same for all IPv4 packets with no options |
| Differentiated service | 1     | Default 0 × 00 common                     |
| Length                 | 2     | Provided by packet characteristics        |
| Identification         | 2     | 0 × 00 0 × 00 for many clients            |
| Flags/frag offset      | 2     | 0 × 00 0 × 00 (no flags, no fragments)    |
| Time to live           | 1     | Commonly used values are 128 and 16       |
| Protocol               | 1     | User datagram protocol (UDP), 0 × 11      |
| Header checksum        | 2     | Calculable for any guessed header         |
| Source address         | 4     | 0.0.0.0 (client yet to have an address)   |
| Destination address    | 4     | 255.255.255.255 (broadcast)               |
| UDP header             |       |   |
| Source port            | 2     | DHCP, 68                                  |
| Destination port       | 2     | DHCP, 67                                  |
| Length                 | 2     | Provided by packet characteristics        |
| Total                  | 34    |   |

## The next step, build up to MTU

- The keystream recovered by the previous process is limited in length. We must try a trial-and error scheme to extend it to a full MTU.
  - An “inductive” attack using a innocuous higher layer protocol – such as ARP (malformed!) and/or ICMP echo (ping). No response means the message we constructed was not valid, if there is a response, it was valid. (One could also try to send datagrams to a known closed port.)

# An inductive attack



N. Petroni & W. Arbaugh, "The Dangers of Mitigating Security Design Flaws: A Wireless Case Study," IEEE Security & Privacy, Jan. 2003.