Nomadic Communications Labs

Alessandro Villani
avillani@science.unitn.it

WEP Cracking

WEP: Wired Equivalent Privacy

- The aim declared of the **WEP** (Wired Equivalent privacy) key is (*Nomina sunt consequentia rerum*) providing *"a security level on the wireless channel equivalent to what one can expect in the case of wired networks"*
- Some have thought WEP as the sole mechanism for the access control
- Other as the solution of all the security problems

WEP: Wired Equivalent Privacy

- The shared key must be installed on the Access Point and on the client
- On the devices up to 4 keys are configurable but the standard does not specify how manage these keys: in the practice just one is used
- PROBLEM:
  - It is not possible install/update it
  - It is the same for all the users of the same AP

WEP: How it works

- WEP is based on the RC4 algorithm of the RSA
- It is a system of encryption based on a shared key
- The shared key is 40 bits (or 104 bits) long
- It is joined with an ***initialization vector*** (IV) 24 bits long
- In this way a seed of 64 bits (or 128 bits) is obtained for the RC4

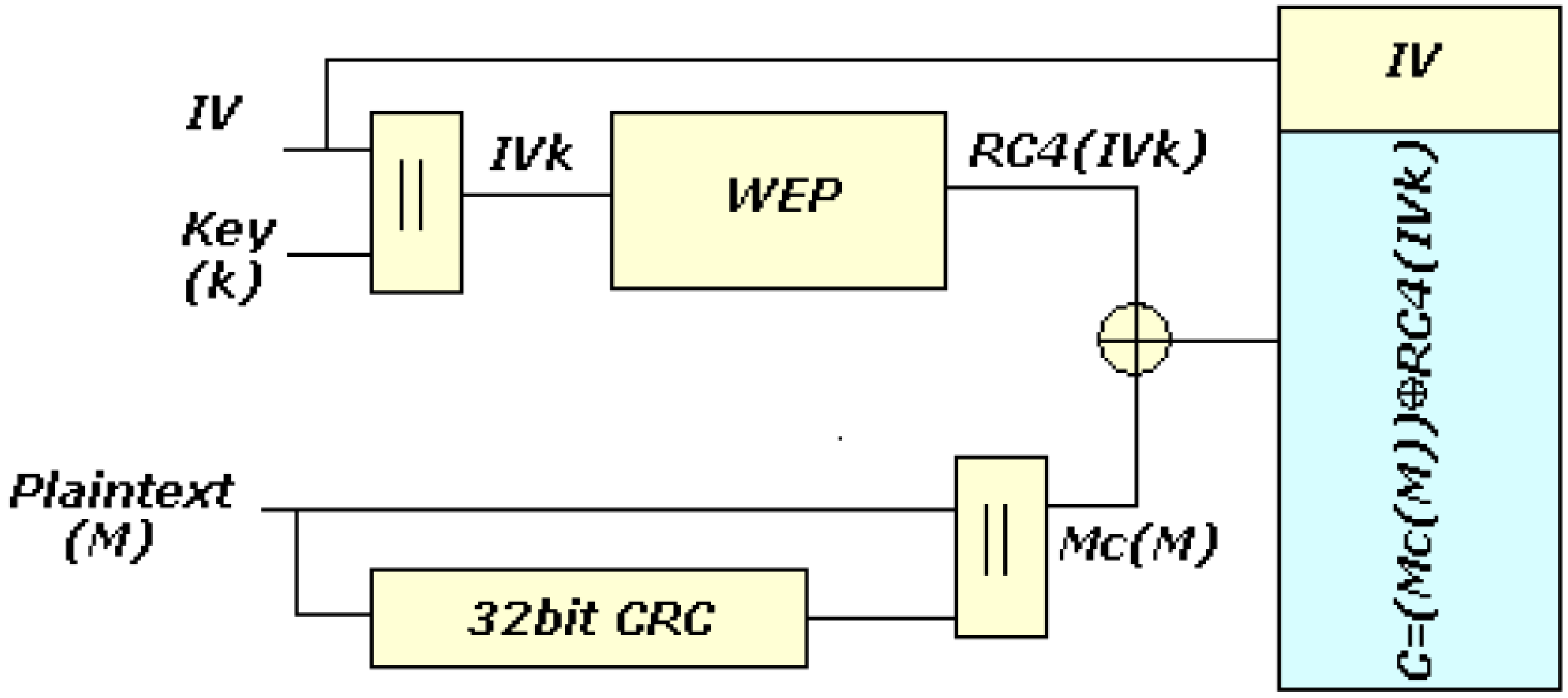
WEP: How it works

- To send a data packet:
  - Given the payload M, the 32 bits CRC c(M) is calculated and concatenated to M → M·c(M)
  - The k key is concatenated to the IV defined for the packet → IV·K
  - The RC4 algorithm is initialized using this packet and a sequence of bytes is produced → RC4(IV·k)
  - Now M·c(M) is xor-ed with RC4(IV·k) → C = (M·c(M))⊕RC4(IV·k)
  - The 3 bytes of the IV are transmitted as clear test (together with the index of the WEP key)

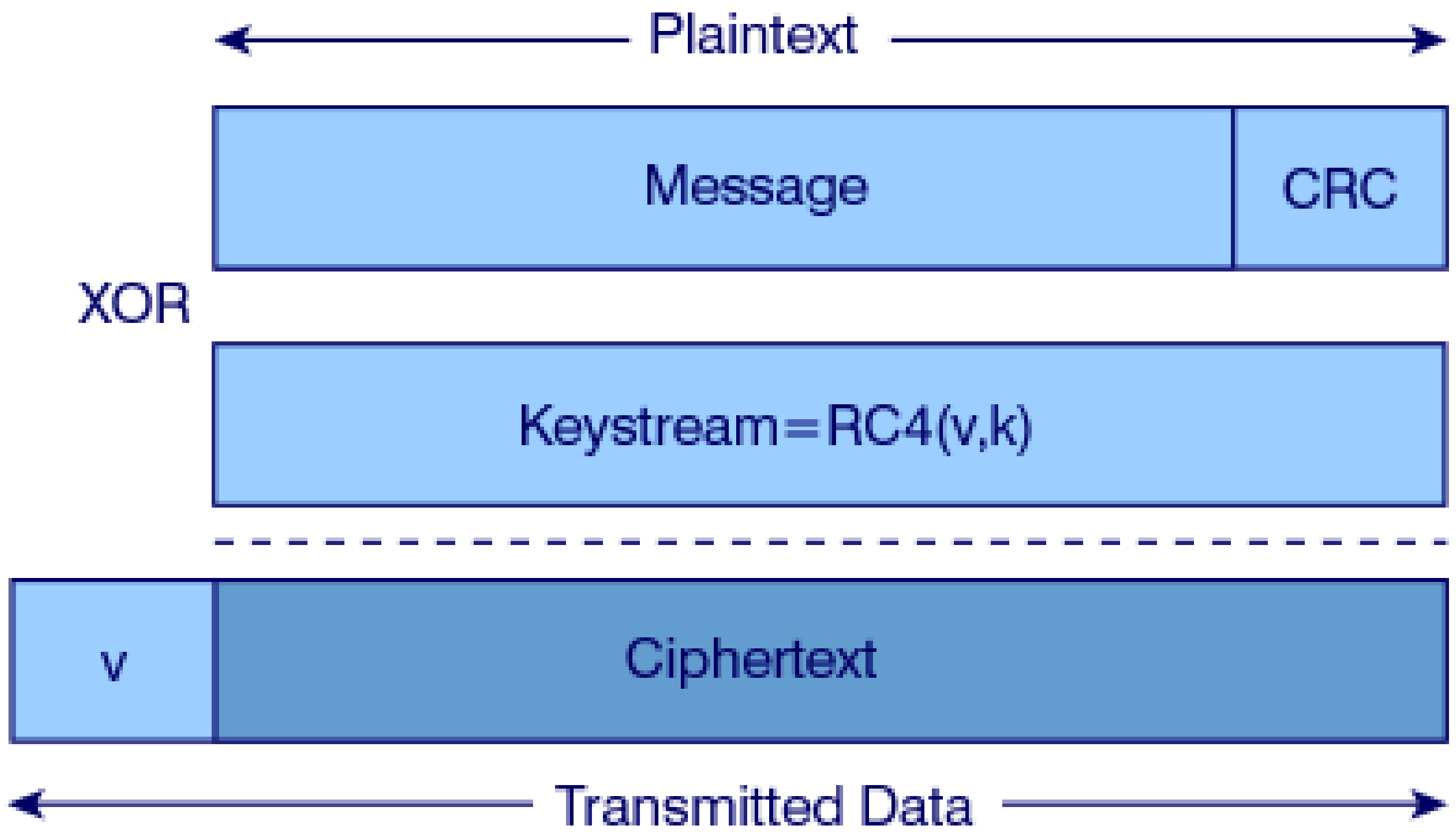
WEP: How it works

- The receiving side concatenates the IV received with the shared WEP key so that it can rebuild RC4(IV·k) → This is the reason for which the IV must be transmitted in clear text
- The receiving side decrypts the payload and if the CRC is equal then the packet is valid otherwise the packet is discarded

WEP: How it works



WEP: How it works



WEP: RC4

- Key Scheduling Algorithm
- RC4 uses a vector of status 256 octets long S[256] and two counters i, j
- Initialization of the status:
  - S [n] = n, i = 0, j = 0
  - In the temporary vector T of 256 octets inserts the IV·K key, repeating it if short
  - S is run exchanging the elements of the vector:

```
 for i = 0 to 255
     j = (j + S[i] + T[i mod 8]) mod 256
     swap (S[i], S[j])
```

WEP: RC4

- Pseudo Random Generation Alghoritm. Generation of the keystream:
  - To generate an octet z of the keystream starting from the actual status (S, i, j):

```
i = (i + 1) mod 256
j = (j + S[i]) mod 256
swap (S[i], S[j])
t = (S[i] + S[j]) mod 256
z = S[t]
```

  - At the beginning i=0, j=0 and discard T
  - The generation process continues until there is no more data

Weakness and Vulnerability of Wep

WEP: Reuse of the coding

- If we use the same IV, the same byte sequence (keystream) is generated from RC4
- Encrypting two messages p1 and p2 we have:
  - $C1 = P1 \oplus RC4(IV \cdot k)$
  - $C2 = P2 \oplus RC4(IV \cdot k)$
  - $C1 \oplus C2 = P1 \oplus RC4(IV \cdot k) \oplus P2 \oplus RC4(IV \cdot k)$
    $= P1 \oplus P2$
- So with the xor of two ciphered messages we get the xor of the two messages as clear text

WEP: Reuse of the coding

- If one of the two messages is known, the other is obtained
- If we have many messages codified with the same keystream it is easy to go back to the original messages
- The protocols impose many similarities to the packets!
- So: do not reuse the keystream
- But: 24 bits of IV means 16.777.216 different keystream: TOO FEW!

WEP: Reuse of the coding

- The standard recommends (but it is not mandatory) that the IV should change in a random way after every transmitted packets
- Some cards generate the 24 bits of the IV using a counter set at zero every time they are initialized and then they increase the counter of 1
  - this increases the probability that the key is reused (the low IV values are more frequent and always transmitted at the beginning of a session)

WEP: Brute Force Attacks

- It can use a list of “easy” keys
- Analyzing the whole research space given
  - Requires up to 45 days with 40 bits
  - Not feasible for 104 bits keys

- Two packets are enough in general (to be sure that the CRC does not coincide by chance also with a wrong WEP key)

WEP: Attacks Based on Weak IV

- S. Fluhrer, I. Mantin, A. Shamir have shown that some weaknesses exist in the algorithm of generation of the keys in RC4 → "*Weakness in the Key Scheduling Algorithm of RC4*"
- The attack described in their article, besides being extremely fast, requires a time which increases linearly with the length of the WEP key!

WEP: Attacks Based on Weak IV

- The fact that a large part of the key (3 bytes) is transmitted in clear and make the cracking easier:
  - The first three iterations of the KSA are easily deducible for the fact that the first three digits of the key are well known (remembered: the IV is transmitted in clear)!
- It is possible to see that there is a probability of 5% than the values in S [0]-S [3] do not change after the first 3 iterations of the KSA

WEP: Attacks Based on Weak IV

- It has been demonstrated that the IV of a certain type are subject to be cracked:

  (B+3:255:x)

  where B is the byte of the secret key (the WEP key) that we are cracking
- Then for every byte of the key there are 256 Weak IV

WEP: Attacks Based on Weak IV

- The first values of the encrypted data is the SNAP (*Sub Network Attachment Point*) header. It is a standard (of layer 2) for the transmission of IP datagram on IEEE 802 network
- The not encrypted header is AA in hexadecimal
- The xor of the first encrypted data with AA, will provide the first byte of the PRGA
- This information allows rebuilding the first digit of the WEP key if we have a Weak IV of the type (3:255:x)

WEP: Attacks Based on Weak IV

- We analyze the first step of the algorithm to produce the first byte of the keystream:

```
i = (i + 1) mod 256 → i = 1
j = (j + S[i]) mod 256 → j = S[1]
swap(S[i], S[j]) → swap(S[1], S[S[1]])
t = (S[i] + S[j]) mod 256 → t = S[1] + S[S[1]]
z = S[t] → z = S[S[1] + S[S[1]]]
```

- So the first byte is function of:
  S[1], S[S[1]] e S[S[1] + S[S[1]]]

WEP: Attacks Based on Weak IV

- The first two steps of the generation of the vector S with IV (3:255:x) are the following:
- i = 0, j = 0

S →	0	1	2	3	4
T →	3	255	x	W_1	

- i = 0, j = (j + S[i] + T[i mod 8]) = (0+S[0]+T[0]) = 0+0+3 = 3 → swap(S[0],S[3])

S →	3	1	2	0	4
T →	3	255	x	W_1	

WEP: Attacks Based on Weak IV

- i = 1, j = 3

S →	3	1	2	0	4
T →	3	255	x	W_1	

- i = 1, j = (j + S[i] + T[i mod 8]) = (3+S[1]+T[1]) = 3+1+255 = 3 → swap(S[1],S[3])

S →	3	0	2	1	4
T →	3	255	x	W_1	

WEP: Attacks Based on Weak IV

- At the next step j = (3 + S[2] + T[2]) = (3 + (2 + x)) that is j move forwards of x + 2 with x known
- Every IV behaves in different way depending on x, but we are able to rebuild the configuration of the vector S
- From here on the evolution of S depends on the key, and with a probability of 5% (as we said previously) the first 3 values of S do not change

WEP: Attacks Based on Weak IV

- Beyond the first byte of the key the operation gets complicated because it requires to go through the PRGA for several steps and so we could not be able to infer with a reasonable probability the exchanges of S
- Also other Weak IV families exist

WEP: Attacks Based on Weak IV

- Some producers of wireless cards have started building cards which avoid IV weak
- The space of IV available is further reduced (some thousands less)
- Observe that, to complete the attack, it is enough that only one client does not avoid the weak IVs

Airsnort: software for the cracking of WEP keys

Airsnort

- Several tools exist which allow to determine in an automatic way a WEP key
- One of these is Airsnort, downloadable to the address:
  http://airsnort.shmoo.com/
- It is a linux program now also for windows
- It requires the wireless card in monitor mode
- It works for instance with the cards Prism2, Orinoco and Cisco

Airsnort

- Once activated, the program captures the packets and simultaneously tries to crack the WEP key:
  - All the non data packets (except the beacon) are dropped
  - The packets not encrypted are dropped
  - The encrypted packets are selected and the ones considered not interesting are dropped
- The packets considered interesting are the Weak IV identified by Fluhrer, Mantin and Shamir (plus several *Weak IV* identified afterwards)

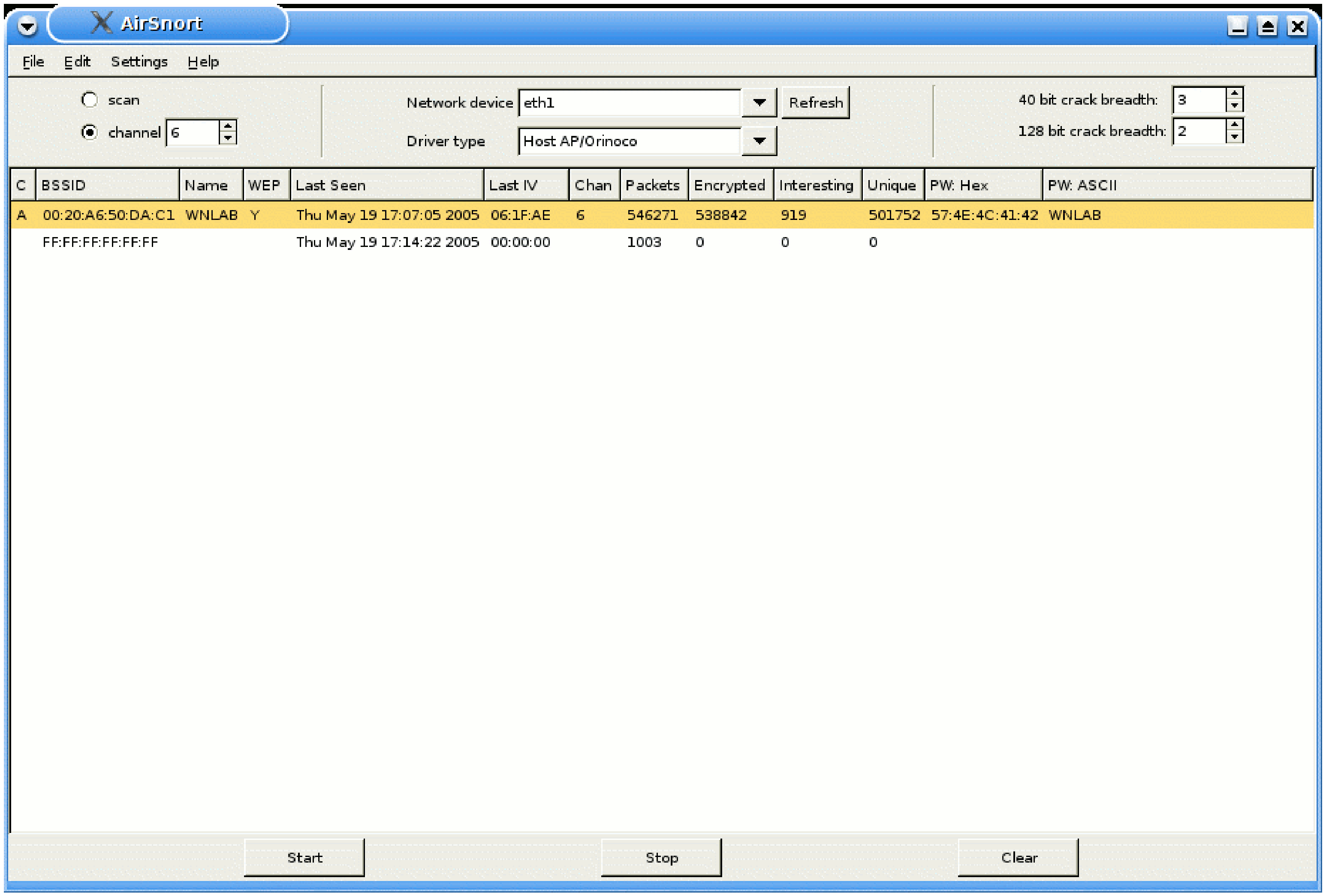
Airsnort

- Every 10 weak IV acquired, airsnort uses a probabilistic attack
- It is possible to define how deep the analysis of the tree of the various possibilities must be
- A value n of the parameter “breadth” indicates that the algorithm will try the n more probable values for each position of the key
- About 1000 weak IV for a key to 64 bits and about 2000 for a key to 128 bits are required

Airsnort

- Test of attack completed using:
  - An Access Point Avaya AP3
  - Two laptop to produce traffic
  - A laptop with a Netgear wireless card and Airsnort
- Set up a 64 bits WEP to, that is 40 key bits, that is 5 characters → WNLAB
- After about 15 minutes of acquisition with about 550.000 packets (540.000 encrypted) and 919 Weak IV, the key has been determined!

Airsnort



( ) scan [✓ channel] 6

Network device: eth1

Driver type: Host AP/Orinoco

40 bit crack breadth: 3

128 bit crack breadth: 2

C	BSSID	Name	WEP	Last Seen	Last IV	Chan	Packets	Encrypted	Interesting	Unique	PW: Hex	PW: ASCII
A	00:20:A6:50:DA:C1	WNLAB	Y	Thu May 19 17:07:05 2005	06:1F:AE	6	546271	538842	919	501752	57:4E:4C:41:42	WNLAB
	FF:FF:FF:FF:FF:FF			Thu May 19 17:14:22 2005	00:00:00		1003	0	0	0		

Airsnort

- In the following table some runs of Airsnort:

Key length	No. of packets	Enchrypted packets	weak IV
40	283618	278860	120
40	546271	538842	919
40	283895	280098	100
40	283876	280057	102
40	702466	676083	252
104	285328	281596	104
104	285798	282076	850
104	575137	567385	933

Airsnort: WEP and IV

- You have to:
  - Acquire some data generated from the laptop you intend to analyze (ping the AP)
  - Take a contiguous sequence of two or more packets
  - Look at the first 3 bytes that are the IV!
  - Restart the network card, to verify if the IV sequence start each time from the same value

WEP and IV: the task

- Many wireless cards generate IV in a predictable way
- Today we will analyze the IV sequences for some wireless cards
- You have to work with two laptop:
  - One where you install the wireless card (verify also the integrated wireless network card)
  - The other in monitor mode to acquire and analyze the traffic