Today

Wi-Fi security

WEP and its problems

WPA and its future

Background

What assumptions do we make on wired networks?
What assumptions *should* we make?

Dolev-Yao assumptions

- protocols should work on untrusted networks!
- often want to control *access* to a network
- link-level privacy not a *bad* idea (defence in depth)

OSI layers
Application
Presentation
Session
Transport
Network
Data link
Physical

Just like Kirchoffs never said that you *have* to give the design of a cipher system to the adversary, Dolev and Yao don't say that we *have* to let our adversary tap into our network routers' core software. Instead, they say that ____
____

IEEE 802.1X

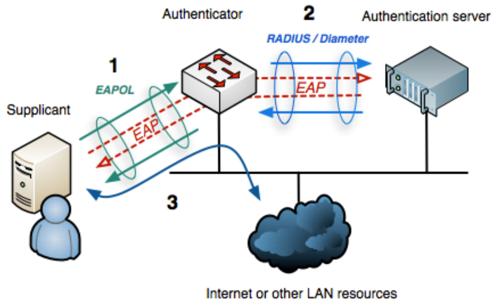
Protected network access

Authentication: who are you?

Authorization: can you use this network?

Audit/accounting: AAA or AAAA via RADIUS… or Diameter

Other policy: ever try to share the Memorial wired network via Wi-Fi?



Source: Arran Cudbard-Bell via Wikimedia

At home, you can set your computer to act as a gateway and pass traffic from Wi-Fi clients through to the wired network. On the Memorial network, however, if you try to set your computer to be a Wi-Fi access point, you'll get a message saying that the network policy doesn't allow you to do that. That's because the wired network uses IEEE 802.1X to announce a "don't share me" policy.

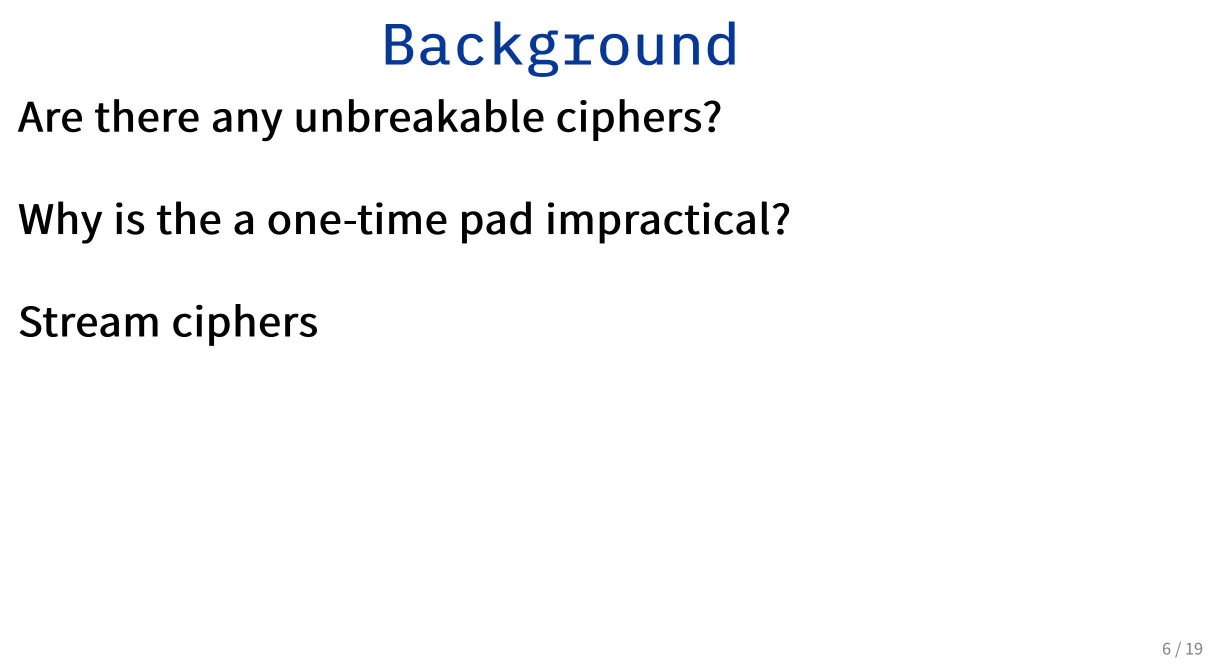
Question: how would that policy be enforced?

History

IEEE 802.11

- 1997: let's make computers talk to each other without wires!
- original plan for security: *wired equivalent privacy* (WEP)
- not exactly successful

Providing equivalent security to wired Ethernet networks wasn't exactly a lofty goal, and yet it's a goal that wasn't reached.



Background

Are there any unbreakable ciphers?

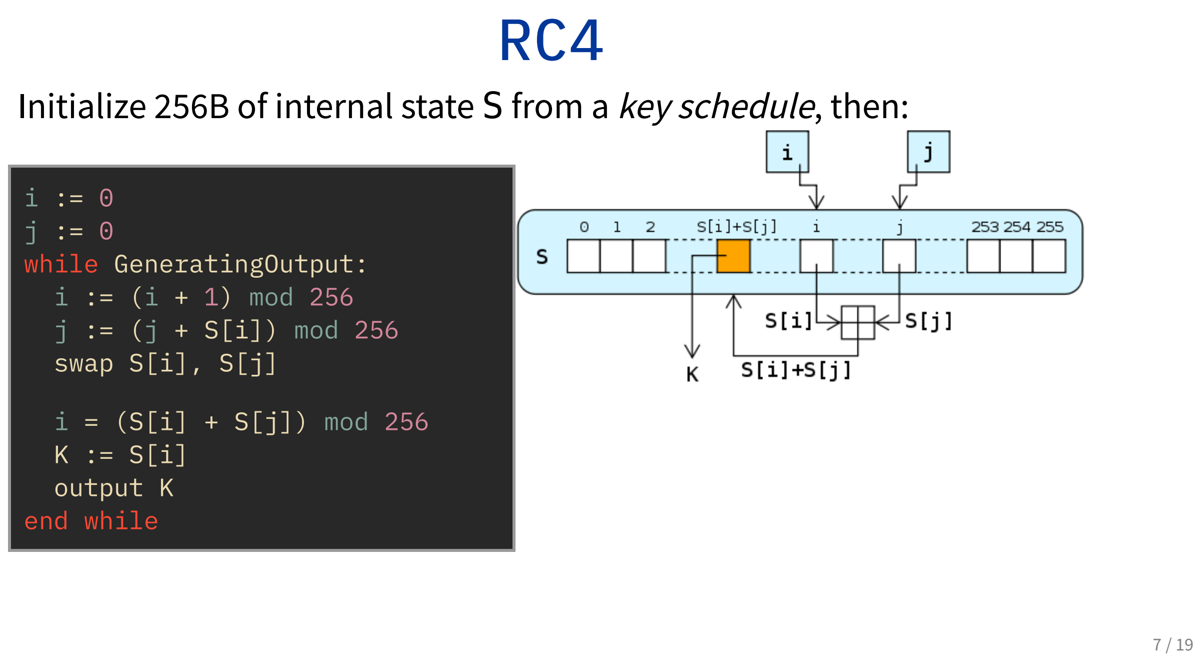
Why is the a one-time pad impractical?

Stream ciphers

We know that the one-time pad, if used with a random keystream, provides ____ ______ ______ _____ (a strong but provable claim!).

The problem with the one-time pad is that we have to ______________________________ __ __________ to our communication endpoints. That's difficult to do without a global network of couriers, etc. So, instead, one thing we can do is try to _____ _ __________________ __ ___________ using a keystream derived from a ____________ ________.

There are lots of good examples of stream ciphers: you can use a block cipher with a stream cipher mode like GCM, or you can use a purpose-designed stream cipher like Salsa20 or Trivium. However, Wi-Fi predates many of these options, so it used...



RC4

Initialize 256B of internal state S from a *key schedule*, then:

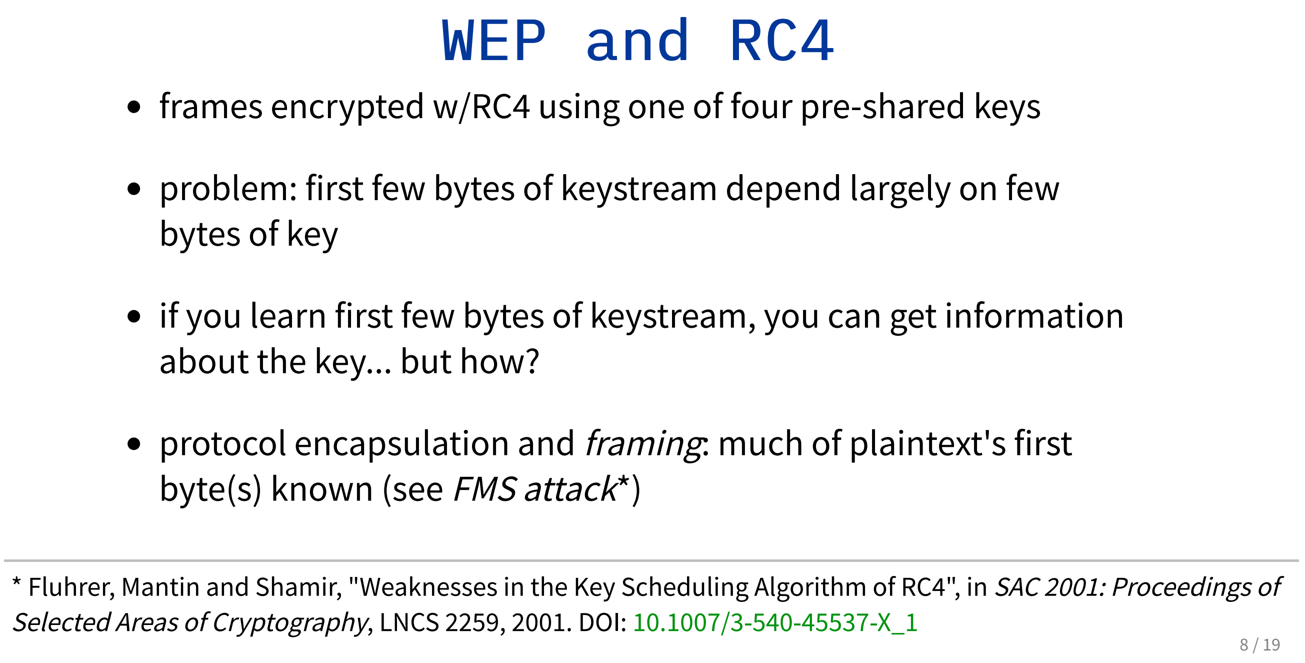
```
i := 0
j := 0
while GeneratingOutput:
  i := (i + 1) mod 256
  j := (j + S[i]) mod 256
  swap S[i], S[j]

  i = (S[i] + S[j]) mod 256
  K := S[i]
  output K
end while
```

RC4 ("Rivest Cipher 4" or "Ron's Code 4") was designed in the 1980s; its design was initially kept secret by RSA Security (where the "R" stands for "Rivest"). This was a state-of-the-art algorithm designed by a famous cryptographer and used by a firm whose entire business is computer security. However, it wasn't subjected to critical external review because it was guarded as a ____________________. Like a lot of secrets, it eventually leaked and people started writing code to generate "ARC4" output (an algorithm that is 100% compatible with RC4 but which didn't suffer from the legal ambiguity associated with RSA's IP).

Once RC4 gets going, it provides confusion via non-linear access to values that come from the current state. It also has diffusion, as values get swapped around within that internal state. So... good?

The problem is that these properties only really apply ________________________________ __________ and ___________________________. The first few bytes of keystream largely depend on ______________________________.



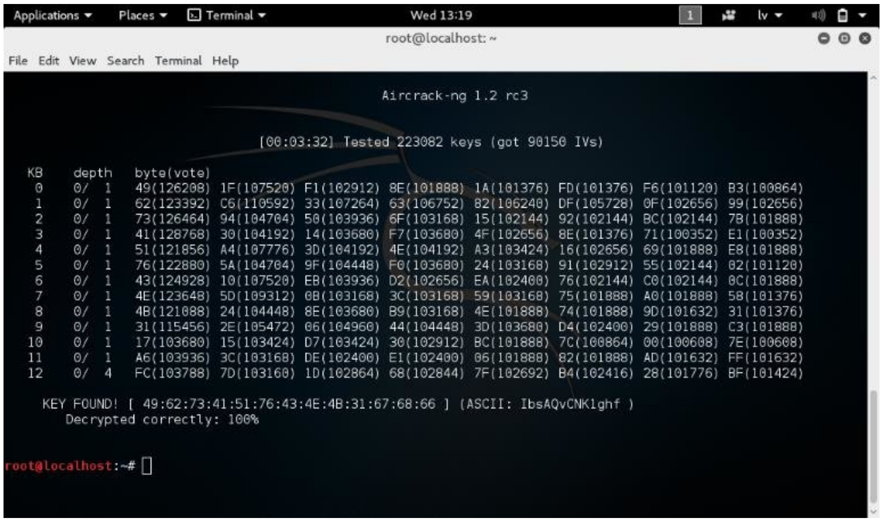
WEP and RC4

- frames encrypted w/RC4 using one of four pre-shared keys
- problem: first few bytes of keystream depend largely on few bytes of key
- if you learn first few bytes of keystream, you can get information about the key... but how?
- protocol encapsulation and *framing*: much of plaintext's first byte(s) known (see *FMS attack**)

* Fluhrer, Mantin and Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4", in *SAC 2001: Proceedings of Selected Areas of Cryptography*, LNCS 2259, 2001. DOI: 10.1007/3-540-45537-X_1

As we've seen in the lab and elsewhere, networking is full of ____________________. For example, an HTTP packet has HTTP traffic (with HTTP headers) contained in a TCP packet (with TCP headers) which is contained in an IP packet (with IP headers) which is contained in a network frame (typically with Ethernet headers). That means that many of the bytes at the beginning of a network packet are ____________________ — or even ____________.

WEP cracking



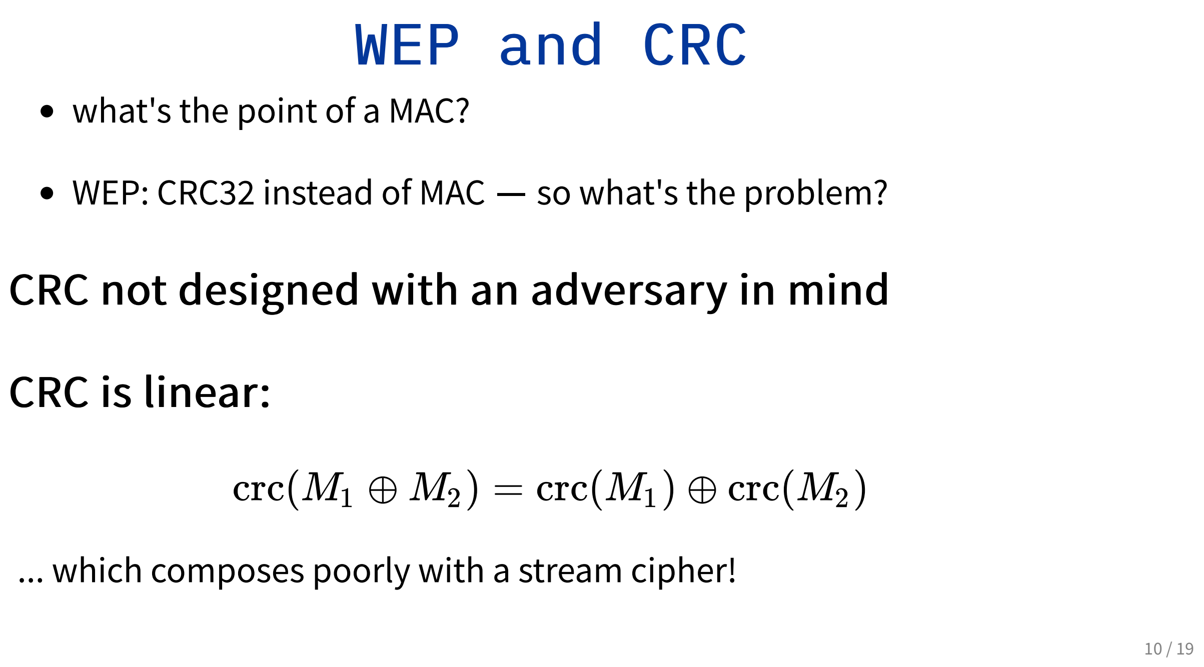
```
                                 Aircrack-ng 1.2 rc3


                 [00:03:32] Tested 223082 keys (got 90150 IVs)

   KB    depth   byte(vote)
    0    0/  1   49(126208) 1F(107520) F1(102912) 8E(101888) 1A(101376) FD(101376) F6(101120) B3(100864)
    1    0/  1   62(123392) C6(110592) 33(107264) 63(106752) 82(106240) DF(105728) 0F(102656) 99(102656)
    2    0/  1   73(126464) 94(104704) 50(103936) 6F(103168) 15(102144) 92(102144) BC(102144) 7B(101888)
    3    0/  1   41(128768) 30(104192) 14(103680) F7(103680) 4F(102656) 8E(101376) 71(100352) E1(100352)
    4    0/  1   51(121856) A4(107776) 3D(104192) 4E(104192) A3(103424) 16(102656) 69(101888) E8(101888)
    5    0/  1   76(122880) 5A(104704) 9F(104448) F0(103680) 24(103168) 91(102912) 55(102144) 02(101120)
    6    0/  1   43(124928) 10(107520) EB(103936) D2(102656) EA(102400) 76(102144) C0(102144) 0C(101888)
    7    0/  1   4E(123648) 5D(109312) 0B(103168) 3C(103168) 59(103168) 75(101888) A0(101888) 58(101376)
    8    0/  1   4B(121088) 24(104448) 8E(103680) B9(103168) 4E(101888) 74(101888) 9D(101632) 31(101376)
    9    0/  1   31(115456) 2E(105472) 06(104960) 44(104448) 3D(103680) D4(102400) 29(101888) C3(101888)
   10    0/  1   17(103680) 15(103424) D7(103424) 30(102912) BC(101888) 7C(100864) 00(100608) 7E(100608)
   11    0/  1   A6(103936) 3C(103168) DE(102400) E1(102400) 06(101888) 82(101888) AD(101632) FF(101632)
   12    0/  4   FC(103788) 7D(103160) 1D(102864) 68(102844) 7F(102692) B4(102416) 28(101776) BF(101424)

     KEY FOUND! [ 49:62:73:41:51:76:43:4E:4B:31:67:68:66 ] (ASCII: IbsAQvCNK1ghf )
        Decrypted correctly: 100%


root@localhost:~# 
```



WEP and CRC

- what's the point of a MAC?
- WEP: CRC32 instead of MAC — so what's the problem?

CRC not designed with an adversary in mind

CRC is linear:

$$\mathrm{crc}(M_1 \oplus M_2) = \mathrm{crc}(M_1) \oplus \mathrm{crc}(M_2)$$

... which composes poorly with a stream cipher!

10 / 19

Because the CRC is linear, I can know what bits of the CRC will change if one bit of the message gets flipped. Put more adversarially, if an attacker Mallory wants to flip a bit in a message, she knows ________________________________ in order to make the CRC continue to match the message. This is very different from a ____________, in which an attacker modifying a message has no way to __________________________________ _______.

This is still true when working with a message and CRC that have been encrypted with a stream ciphier (or one-time pad!). An attacker can flip ciphertext bits and "fix up" the CRC without needing to know ______________________________, only that they are flipping.

WEP integrity

Not much!

So in WEP, an attacker could alter arbitrary bits within an encrypted packet and have the packet still accepted as valid. This is not good, especially given that we know where lots of data is located within ciphertext packets due to _____________________. For example, suppose you wanted to alter a destination IP address to cause a Wi-Fi client to send data to you instead of its intended destination... if that destination address is well-known, you can!

ChopChop attack

It gets worse!

Wi-Fi APs as oracles

Chopping off bytes

If you send a packet to an access point whose CRC doesn't match the message, the AP will respond with an error frame that is different from other errors. This allows an attacker to treat the AP as an oracle that distinguishes between "good CRC" and "bad CRC". If you take a previously-broadcast packet and retransmit *most* of it, you can ask the AP "is this CRC good?"

This doesn't seem like much of a superpower until you remember that the relationship of the CRC to the message is linear, so it's possible to construct linear relationships among the bytes of the original frame, the shortened frame, the CRC bytes ______________________________. By trying variations on the shortened frame's CRC, you can ______________________________ ________________. If you repeat this process, you can ________________________________.

WPA: Wi-Fi Protected Access

(stopgap, partial implementation of 802.11i)

TKIP: Temporal Key Integrity Protocol

- still uses RC4 (compatibility with old hardware)
- better key scheduling (not just key+IV) stops *related key attacks*
- new encryption key every packet

Message Integrity Code (MIC)

WPA2

IEEE 802.11i-2004

Mandatory AES-CCMP

PSK or EAP

EAP (Extensible Authentication Protocol) uses the same RADIUS infrastructure as IEEE 802.1X, so you can plug it into your existing enterprise authentication infrastructure.

WPA-PSK

Four-way handshake:

$$
\begin{aligned}
&AP \rightarrow S : N_{AP} \\
&S \rightarrow AP : N_S, MIC \\
&AP \rightarrow S : \{k_{GTK}, MIC\}_{k_{PTK}} \\
&S \rightarrow AP : MIC
\end{aligned}
$$

where $k_{PTK} = h\left(k_{PMK} + N_{AP} + N_S + MAC_{AP} + MAC_S\right)$
... from which **tons** of other keys are derived

The GTK (Group Temporal Key) is the key that's generally used for encryption on the wireless network. It can be changed periodically (it's a *temporal* key).

The Pairwise Temporal Key is used for this initial pairwise communication between the *station* and the access point. It is derived from the Pairwise Master Key, which can be derived from a Pre-Shared Key (PSK) if there is "a passphrase" for the network, or can come from RADIUS in the case of EAP (see next slide). Note that "MAC" in this instance means "Ethernet MAC address", *not* Message Authentication Code!

WPA-EAP

EAP: Extensible Authentication Protocol

- Ethernet frame protocol defined by IEEE 802.1X (even wired)
- Allows authentication to be tunnelled to an external AAA server (e.g., RADIUS)

Memorial: EAP-PEAP w/MSCHAPv2

Modern WPA

Moving to WPA3

- forward secrecy!
- *zero-knowledge* mutual authentication via PAKE
    - PAKE: *password-authenticated key exchange*
      SAE: *simultaneous authentication of equals* (IEEE 802.11s)
    - **Diffie-Hellman** (or, rather, a variant of it) strikes again!

But the game goes on...

There are a variety of PAKE protocols out there, all of which allow two parties to use a password to prevent middleperson attacks in something like Diffie-Hellman key exchange.

Simultaneous Authentication of Equals is a protocol that also allows two parties to establish a key while proving to each other that they both know a pre-shared password (but without having to reveal any information about that password).

The attack against the WPA3 handshake is one of those vulnerabilities that has to have a cute name ("Dragonblood") and a shiny website.

Today

Wi-Fi security

WEP and its problems

WPA and its future