Cryptographic Engineering

- We started talking about cryptography last time (certificates)
- How to use cryptography properly isn't obvious
- Must lay a bit of introductory groundwork

What is a Cryptosystem?

- $K = \{0,1\}^l$
- $P = \{0,1\}^m$
- $C' = \{0,1\}^n, C \subseteq C'$
- $E : P \times K \rightarrow C$
- $D : C \times K \rightarrow P$
- $\forall p \in P, k \in K : D(E(p,k),k) = p$
- It is infeasible to find $F : P \times C \rightarrow K$

Let's start again, in English. . .

What is a Cryptosystem?

A cryptosystem is pair of algorithms that take a *key* and convert *plaintext* to *ciphertext* and back.

Plaintext is what you want to protect; ciphertext should appear to be random gibberish.

The design and analysis of today's cryptographic algorithms is highly mathematical. Do *not* try to design your own algorithms.

Properties of a Good Cryptosystem

- There should be no way short of enumerating all possible keys to find the key from any reasonable amount of ciphertext and plaintext, nor any way to produce plaintext from ciphertext without the key.
- Enumerating all possible keys must be infeasible.
- The ciphertext must be indistinguishable from true random values.

Kerckhoffs' Law (1883)

> The system must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.

In other words, the security of the system must rest entirely on the secrecy of the key.

Objectives for the User

- Generate good (i.e., unguessable) keys
- Protect keys from compromise
- Prevent attacker from getting system to do the decryption for us

What We Have Today

- Encryption is completely computerized, and operates on bits
- The basic primitives of encryption are combined to produce very powerful results
- Encryption is by far the strongest weapon in the computer security arsenal; host and operating system software is by far the weakest link
- *Bad software trumps good crypto*

Key Management

- Where do cryptographic keys come from?
- Generally, each party has (at least) one long-term key
- Some procedure or protocol is followed to produce a traffic key or *session key*
- Avoids too much exposure of the long-term key

Block Ciphers

- Operate on a fixed-length set of bits
- Output blocksize generally the same as input blocksize
- Well-known examples: DES (56-bit keys; 64-bit blocksize); AES (128-, 192-, and 256-bit keys; 128-bit blocksize)

Cipher Strengths

- A cipher is no stronger than its key length: if there are too few keys, an attacker can enumerate all possible keys
- DES has 56 bits — arguably too few in 1976; far too few today.
- Strength of cipher depends on how long it needs to resist attack.
- No good reason to use less than 128 bits
- NSA rates 128-bit AES as good enough for SECRET traffic; 256-bit AES is good enough for TOP-SECRET traffic.
- But a cipher can be considerably weaker! (A monoalphabetic cipher over all bytes has a $256! = 1684$-bit key, but is trivially solvable.)

Brute-Force Attacks

- Try every possible key
- First proposed for DES by Diffie and Hellman (1979); then, the costs were such that only major governments could do it
- Assumptions: 1,000,000 chips; each tries 1,000,000,000 keys/second
- $2^{56}/(10^6 \cdot 10^9)$ is 72 seconds
- Those parameters are a stretch, but a DES-cracker was built, with private funds, in 1997: $250,000 machine; cracked DES in 4 days
- AES: $2^{128}/(10^6 \cdot 10^9) = 72 \cdot 2^{72}$ seconds = 10^{16} years
- Not in my threat model. . .

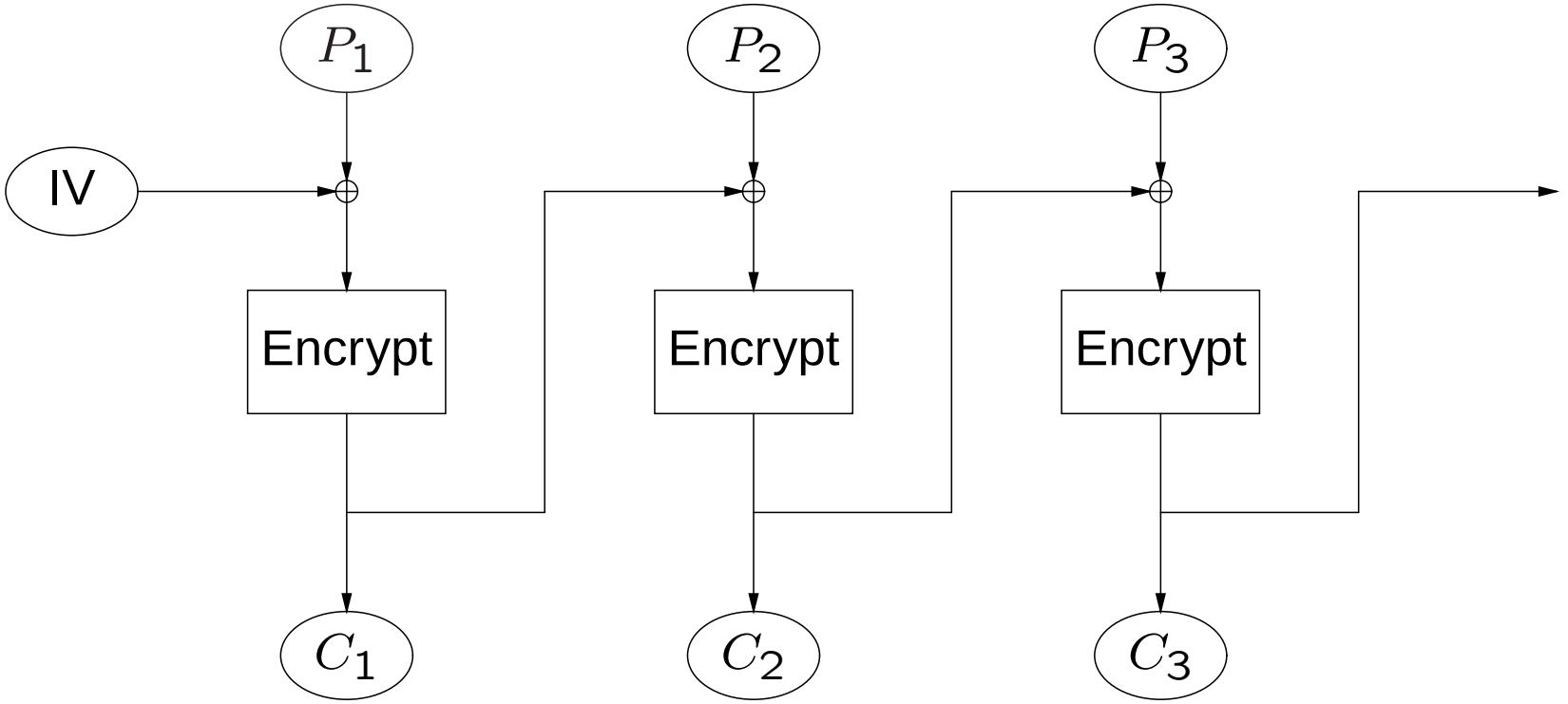
Modes of Operation

- Direct use of a block cipher is inadvisable
- Enemy can build up “code book” of plaintext/ciphertext equivalents
- Beyond that, direct use only works on messages that are a multiple of the cipher block size in length
- Solution: five standard *Modes of Operation*: Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR).

Electronic Code Book

- Direct use of the block cipher
- Used primarily to transmit encrypted keys
- Very weak if used for general-purpose encryption; never use it for a file or a message.
- We write $\{P\}_k \rightarrow C$ to denote "encryption of plaintext P with key k to produce ciphertext C"

Cipher Block Chaining



$$\{P_i \oplus C_{i-1}\}_k \rightarrow C_i$$
$$\{C_i\}_{k^{-1}} \oplus C_{i-1} \rightarrow P_i$$

Properties of CBC

- The ciphertext of each encrypted block depends on the plaintext of all preceeding blocks.
- There is a dummy initial ciphertext block C_0 known as the *Initialization Vector* (IV); the receiver must know this value.
- Consider a 4-block message:

$$\begin{aligned} C_1 &= \{P_1 \oplus IV\}_k \\ C_2 &= \{P_2 \oplus C_1\}_k \\ C_3 &= \{P_3 \oplus C_2\}_k \\ C_4 &= \{P_4 \oplus C_3\}_k \end{aligned}$$

  If C_2 is damaged during transmission, what happens to the plaintext?

Error Propagation in CBC Mode

- Look at the decryption process, where C' is a garbled version of C:

$$\begin{aligned} P_1 &= \{C_1\}_{k^{-1}} \oplus IV \\ P_2 &= \{C_2'\}_{k^{-1}} \oplus C_1 \\ P_3 &= \{C_3\}_{k^{-1}} \oplus C_2' \\ P_4 &= \{C_4\}_{k^{-1}} \oplus C_3 \end{aligned}$$

- P_1 depends only on C_1 and IV, and is unaffected
- P_2 depends on C_2 and C_1, and hence is garbled
- P_3 depends on C_3 and C_2, and is also garbled. The enemy can control the change to P_3.
- P_4 depends on C_4 and C_3, and not C_2; it thus isn't affected.
- Conclusion: Two blocks change, one of them predicatably

Cutting and Pasting CBC Messages

- Consider the encrypted message

$$IV, C_1, C_2, C_3, C_4, C_5$$

- The shortened message IV, C_1, C_2, C_3, C_4 appears valid
- The truncated message C_2, C_3, C_4, C_5 is valid: C_2 acts as the IV.
- Even C_2, C_3, C_4 is valid, and will decrypt properly.
- Any subset of a CBC message will decrypt cleanly.
- If we snip out blocks, leaving IV, C_1, C_4, C_5, we only garble one block of plaintext.
- Conclusion: if you want message integrity, you have to do it yourself.

Stream Ciphers

- Key stream generator produces a sequence S of pseudo-random bytes; key stream bytes are combined (generally via XOR) with plaintext bytes
- $P_i \oplus S_i \rightarrow C_i$
- Stream ciphers are very good for asynchronous traffic
- Best-known stream cipher is RC4; commonly used with SSL
- Key stream S must *never* be reused for different plaintexts:

$$\begin{aligned} C &= A \oplus K \\ C' &= B \oplus K \\ C \oplus C' &= A \oplus K \oplus B \oplus K \\ &= A \oplus B \end{aligned}$$

- Guess at A and see if B makes sense; repeat for subsequent bytes

RC4

- Extremely efficient
- After key setup, it just produces a key stream
- No way to resynchronize except by rekeying and starting over
- Internal state is a 256-byte array plus two integers
- Note: weaknesses if used in ways other than as a stream cipher.

The Guts of RC4

```
for(counter = 0; counter < buffer_len; counter ++)
{
     x = (x + 1) % 256;
     y = (state[x] + y) % 256;
     swap_byte(&state[x], &state[y]);
     xorIndex = (state[x] + state[y]) % 256;
     buffer_ptr[counter] ^= state[xorIndex];
 }
```

Vernam/Mauborgne Cipher (1917-1918)

- Exclusive-OR a key stream tape with the plaintext
- Online encryption of teletype traffic, combined with transmission
- For a one-time pad — which is provably secure — use true-random keying tapes and *never* reuse the keying material.
- If keying material is reusable, it's called a *stream cipher*

☞ Snake oil alert! *If the key stream is algorithmically generated, it's not a one-time pad!*

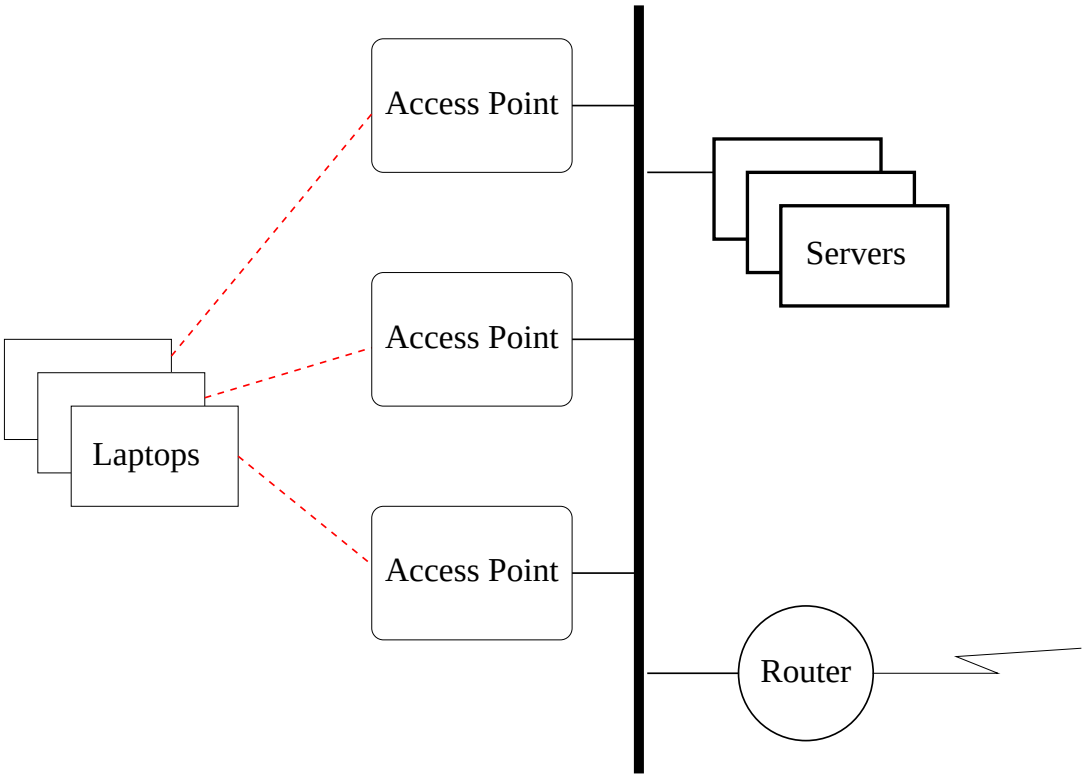
Attacking WEP

- WEP — Wireline-Equivalent Privacy — is the basic security mechanism for 802.11 (WiFi) networks
- It uses RC4 for confidentiality
- Many flaws. . .
- Let's look at two

No Key Management

- WEP has no key management mechanisms
- All users at a site share the same, static key
- No protection against each other (that's why Columbia doesn't use WEP)
- Must change all users' keys simultaneously
- No practical way to recover from a stolen laptop or a disgruntled (ex-) employee

WEP Topology



Only the wireless link is encrypted; decryption happens at the access points

Stealing a Password

- Assume the laptop is talking to a server
- Figure out its IP address D
- Capture an encrypted packet
- Let X be the IP address of the bad guy's machine
- Calculate $X_D = DM \oplus X$
- Exclusive-OR the destination IP address D of the packet with X_D

$$\begin{aligned} D' &= D \oplus X_D \\ &= D \oplus (D \oplus X) \\ &= X \end{aligned}$$

- Reinject the packet
- The access point will decrypt the packet for us, and send it to X

What Was the Mistake?

- We can't prevent access to the ciphertext
- We can't prevent reinjection of new ciphertext
- Stream ciphers permit *predictable, controllable* changes to the plaintext
- Virtually everything encrypted with a stream cipher must contain an *integrity check* as well
- There are many attacks possible if integrity-checking is omitted; this is just one of them. (Block ciphers are vulnerable, too.)

Encrypted File Systems

- We wish to encrypt part of a file system
- How do we do it?
- What are the issues?

Why Encrypt Files?

- Theft of files
- Theft of backup media
- Theft of computer

Bad Reasons and Good

- Is there a flaw in the operating system's protection mechanisms? Why can't the OS keep bad guys from the file?
- You don't trust the system administrator? Can the sysadmin steal the decryption key?

  ☞But — if you're using NFS, the file may reside on one (untrustworthy) machine, while the decryption is done on another
- Laptops have feet — a remarkably high percentage are stolen

Laptop Theft

September 17, 2000

IRVINE – Qualcomm founder Irwin Jacobs' laptop computer disappeared during a conference yesterday in an apparent theft that could put some of the company's most sensitive secrets at risk.

. . .

Jacobs said his laptop contained "everything," secret corporate information, including e-mail dating back years, financial statements and even personal mementos.

. . .

Though Jacobs' IBM ThinkPad PC is valued at about $3,700, the value of the information it contained is incalculable to Qualcomm and to Jacobs.

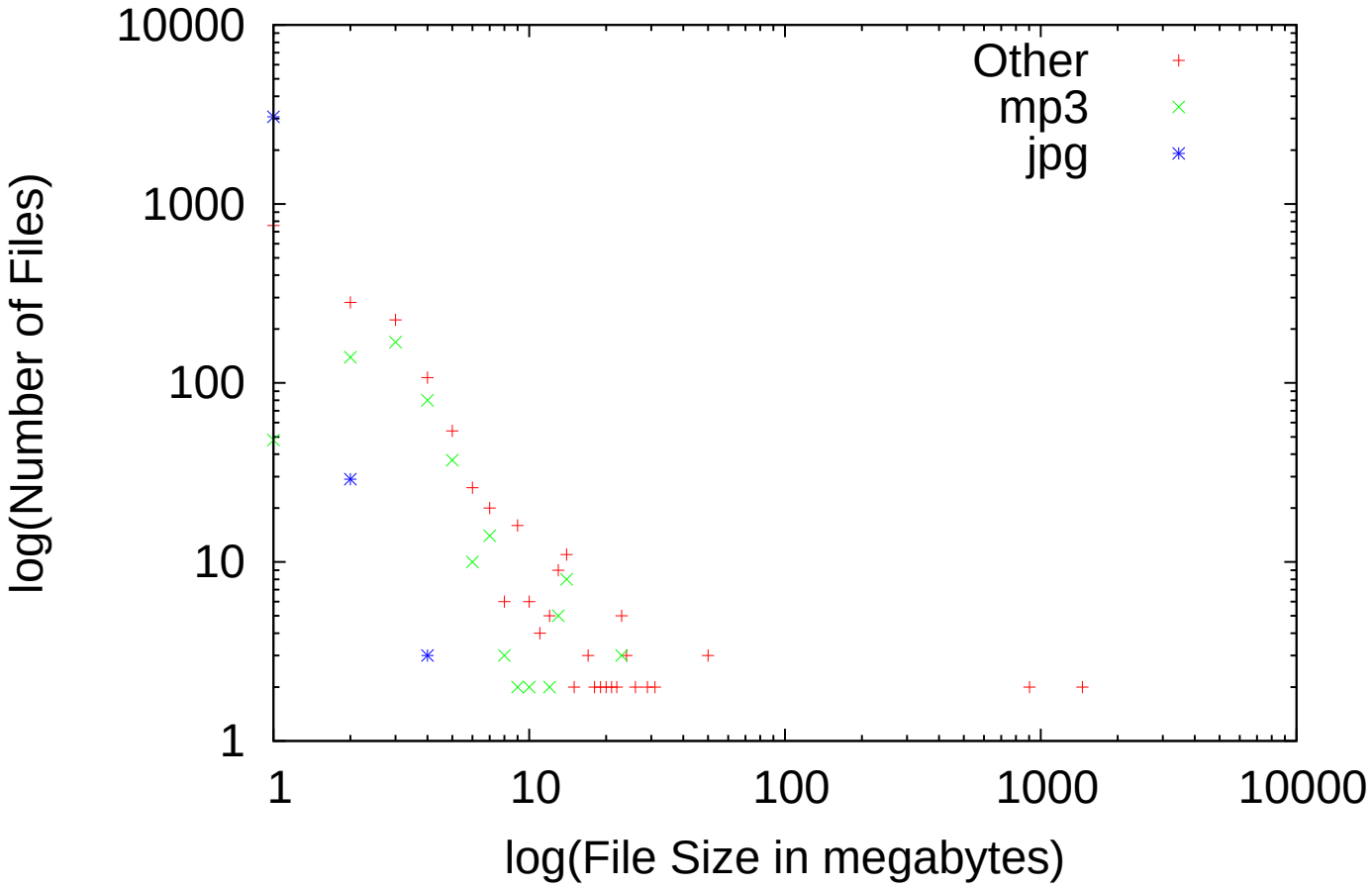
Caveats

- Encrypting a file system provides confidentiality
- It generally does not provide integrity protection
- It may result in a *loss* of availability, if you lose the key

Problems With File System Encryption

- Metadata is not encrypted
- File lengths are not protected
- File name lengths are not well-protected

File Size Distribution



Encryption Options

- Manually encrypt/decrypt files
- Overlay encryption on top of the file system
- Encrypt an entire disk partition

Manual Encryption

- Very inconvenient to use
- Users are constantly supplying keys
- Most utilities won't have direct interfaces to the decryption function; you have to manually decrypt files before use
- Users *will* forget to re-encrypt files

File System Encryption

- Some sort of overlay on real file system
- Encryption and decryption operate on individual files, but transparently to applications
- Directories are files, too, so filenames are encrypted

Encryption Using CFS

```
$ cattach /usr/mab/secrets matt
Key:
$ ls -ld /crypt/matt
drwx------ 2 mab 512 Apr 1 15:56 matt
$ echo "murder" > /crypt/matt/crimes
$ ls -l /crypt/matt
total 1
-rw-rw-r-- 1 mab 7 Apr 1 15:57 crimes
$ cat /crypt/matt/crimes
murder
$ ls -l /usr/mab/secrets
total 1
-rw-rw-r-- 1 mab 15 Apr 1 15:57 8b06e85b87091124
$ cat -v /usr/mab/secrets/8b06e85b87091124
M-Z,k\x{02C6}]\x{02C6}B\x{02C6}VM-VM-6A\x{02DC}uM-LM-_M-DM-\x
```

Doing the Encryption

- What mode of operation do you use?
  ☞CBC is a good choice
- Where does the IV come from? (Note: on Unix, must support seeks to any byte)
- Partial solution: encrypt each block separately; use block number as part of IV
- Must use some metafile for the rest of the IV. Solution must survive file copies, dump/restore, etc. (CFS uses .pvect files.)
- What about never-written blocks? On Unix, these read as all 0s

Disk Encryption

- Encrypt an entire disk or disk partition
- Protects everything, even the free space
- ☞ Very important, given that “delete” operations do not delete the data
- Useful for protecting swap area
- But — free space in encrypted section is not available for plaintext use, and vice-versa

Providing Keys for Encrypted File Systems

- File system encryption: can be supplied by user
- Can have fine-granularity keying, per sub-tree
- Disk Encryption: one key per encrypted partition. Shared?
- In either case, once the key is supplied, you rely on OS protection mechanisms
- Bottom line: file system or disk encryption is useful if the threat is compromise from outside the boundaries of the machine: physical theft, remote file system, backup media, etc.
- It is not useful for intra-machine threats; an enemy who can bypass access controls can steal the key or the plaintext
- Encryption is not a substitute for operating system access controls