Thwart statistical analysis

Shannon in the 1940's suggested two methods:

- Diffusion
  - make statistical analysis hard: spread statistical structure of plaintext in long–range statistics of ciphertext
  - each plaintext bit affect many ciphertext bits
  - ex: permutation + function
- Confusion
  - make key breaking harder: make relation between ciphertext statistics and key value complex
  - ex: complex substitution algorithms

Feistel networks

- Shannons ideas used by Feistel (1970's) – basic structure used since then.
- Product cipher alternating substitution and permutation
  - $c = E_k(m) = S_n \circ P_{n-1} \circ \cdots \circ S_2 \circ P_1 \circ S_1(m)$
- Feistel network
  - split input in two halves L_0, R_0
  - perform n rounds:
    - $F(R_i, k_i) \oplus L_i$
    - swap halves
  - end with a swap

Feistel decryption

- Same algorithm, but keys in reverse order – works independently of *F*

$$LE_{16} = RE_{15} = RD_0 = LD_1 = RE_{15}$$

$$RE_{16} = LE_{15} \oplus F(RE_{15}, K_{16})$$

$$\begin{aligned}
RD_1 &= LD_0 \oplus F(RD_0, K_{16}) \\
&= RE_{16} \oplus F(RE_{15}, K_{16}) \\
&= (LE_{15} \oplus F(RE_{15}, K_{16})) \oplus F(RE_{15}, K_{16}) \\
&= LE_{15} \oplus (F(RE_{15}, K_{16}) \oplus F(RE_{15}, K_{16})) = LE_{15} \oplus 0 \\
&= LE_{15} \\
&\vdots
\end{aligned}$$

$$RD_{16} = LE_0$$

$$LD_{16} = RE_0$$

Feistel net parameters

- Block size (64 bits)
  - larger $\Rightarrow$ greater security (diffusion), but slower
- Key size (128 bits)
  - same relation
- Number of rounds (16)
  - one is too little, more increase security, to a limit
- Subkey generation
  - should be complex
- F should also be complex

Feistel features

- Fast implementation
  - both in software and in hardware
- Can be easy to analyse
  - clear explanation $\Rightarrow$ easier to analyse $\Rightarrow$ safer to trust
  - (DES is not easy to analyse)

Data Encryption Standard (1977)

- Most common variant of a Feistel net
- Encrypts 64–bit blocks with 56–bit key
- Hardware implementations (in USA)
- Known and much analysed algorithm
  - export control on implementations (earlier)
  - unknown criteria for design
    - unknown if trap doors exist

Breaking DES by brute force

- 1977: estimated breakable in 1 day by $20M machine
- 1981: estimated breakable in 2 days by $50M machine
- 1997: broken in 96 days by 70,000 machines, testing 7 billion keys/sec
- 1998: less than 3 days by special hardware, $250K incl design & development
- 1999: in 22h15m, "Deep Crack" + 100,000 machines, testing 245 billion keys/sec

Key generation

- Each round uses different keys K_i based on K (64 bits, discard parity bits $\Rightarrow$ 56 bits)
- PC1 permutes and discards parity bits
- Split in two halves C_0,D_0 (28 bits each)
- Each round: C_i = $\mathrm{LS}_i(C_{i-1})$, D_i = $\mathrm{Ls}_i(D_{i-1})$
  - LS_i: left circular shift <1,1,2,2,...,2,2,1> bits
  - $K_i = \mathrm{PC2}(C_i D_i)$

Properties of DES

- Decryption like Feistel (keys in reverse order)
- Symmetry:
  - $c = \text{DES}(m,k)$ iff $\underline{c} = \text{DES}(\underline{m},\underline{k})$ where $\underline{x}$ is x bitwise negated
  - cuts search space in half
- Weak keys
  - cause involution ($E_k(E_k(m)) = m$)
  - 4 exist for DES: (0,0); (–1,0); (0,–1); (–1,–1)
- Semi–weak key pairs
  - if $E_{k1}(E_{k2}(m)) = m$
  - 6 such pairs exist for DES (few enough to check for)

Avalanche effect

- Small changes in *m* or *k* give big changes in *c*, and the changes increase for each round
- Ex: one bit change in plaintext or key:

Change in plaintext		**Change in key**	
Round	**Bits differ**	**Round**	**Bits differ**
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
14	26	14	26
15	29	15	34
16	34	16	35

Design criteria

- S–box design
  - very careful for DES (some properties in sec. 3.6)
  - can in general be done
    - randomly
    - randomly with testing
    - by careful hand–crafting
    - mathematically
- Number of rounds
  - brute force requires 2^{55} tests
  - for DES with 16 rounds, *differential cryptanalysis* requires $2^{55.1}$ operations
  - with 15 rounds, diffrential c.a. would beat brute force

Design criteria (cont)

- Function *F*
  - Strict Avalanche Criterion
    - any output bit changes with p=½ if a single input bit changes
  - Bit Independence Criterion
    - any two output bits should change independently when a single input bit changes

Strengthening DES

- Double DES
  - $c = E_{k2}(E_{k1}(m))$
- Avoid idempotence ($=E_{k3}(m)$)
  - unlikely: $2^{64!}$ mappings from M to C possible, but only 2^{56} different keys possible
    - low probability for two keys to give same mapping as one
  - proven impossible in 1992
- Meet–in–the–middle attack
  - $c = E_{k2}(E_{k1}(m)) \Rightarrow E_{k1}(m) = D_{k2}(c)$
  - known plaintext, two cases $\Rightarrow$ very likely to find correct key (but requires 2^{56} tests: double to DES)

Triple DES

- Two keys: $c = E_{k1}(D_{k2}(E_{k1}(m)))$
  - cost of known–plaintext attack: 2^{112}
  - D in the middle for backwards compatibility:
    - $E_{k1}(D_{k1}(E_{k1}(m))) = E_{k1}(m)$
  - very difficult to break
- Three keys: $c = E_{k3}(D_{k2}(E_{k1}(m)))$
  - used e.g. by PGP

Properties of modern ciphers

Modern ciphers: IDEA, Blowfish, RC5, CAST,...

- Variable key length
- Mixed operations (not only xor, not distr/assoc)
- Data dependent rotations instead of S−boxes
- Key dependent rotations, S−boxes
- Variable F, block length, number of rounds
- Operations on both halves

but basically just improvements of Feistel nets!

Usage modes of block ciphers

- ECB: Electronic Code Book mode
  - plaintext split in (64–bit) blocks
  - each block encrypted separately with same key
  - decryption as usual
  - repetitions in plaintext give repetitions in ciphertext
  - blocks can be swapped, repeated, replaced without noticing

Usage modes (cont)

- CBC: Cipher Block Chaining
  - next plaintext block is xored with previous cipher
  - same key for each block
  - decryption: next plaintext xored with prev. cipher
  - first block xored with Initialization Vector (secret)
  - repetitions do not show up in cipher
  - modifications are detected: each cipher block depends on all previous ones

Modes (cont)

- CFB: Cipher Feedback Mode
  - encrypt j bits at a time: stream cipher
  - encrypt a shift register (initially IV), use j most significant bits xor $m \Rightarrow c$
  - next: shift j bits, inserting previous c, continue

Modes (last)

- OFB: Output Feedback Mode
  - do feedback before xor
  - transmission errors do not propagate
  - more vulnerable to message stream modification
    - changing a cipher bit changes the corresponding plaintext bit
    - change both data and checksum bits $\Rightarrow$ undetected