Block Encryption

Changing a single bit of input only affects 8 bits of output after first round, but potentially all after two.
DES
- Block cipher: uses 56 bit keys, 64 bits of data
- Private-key (symmetric) encryption
- Created by IBM, secret process, probably created to be broken by NSA and made to be inefficient to solve in software (too late, already broken)

64-bit input

64-bit permute

rounds 1-16

2x32-bit swap

64-bit permute

64-bit input

One round of DES

64-bit input

32-bit L_n

32-bit R_n

mangler

key

XOR

32-bit L_{n+1}

32-bit R_{n+1}

64-bit output
The Mangler

56-bit key used to generate 16 per-round 48-bit keys
  • different subsets

  • Input 32 bits broken into 8 4-bit quantities
    • Each expanded to 6 bits by including adjacent bits
    • Each 6-bits xor'd w/ corresponding chunk from $K_n$

  • Each 6-bits input to an S-box, w/ output 4 bits

What's Complicated?

  • Why initial and final permutations
    • Can't be for security (reduction argument, assume <p,c> pair)

  • Why these specific s-boxes

  • Secret design process => suspicion and mistrust.
Concerns about DES

- Short key length
  - Computation can be distributed to make it faster
  - Does not mean “DES is insecure”; depends on desired security
  - How long?
    - Assume we can check a key in 10 ops, comp can do 10^{15}/sec
    - 10^{14} \sim 2^{46.5} \Rightarrow DES \text{ takes } 2^{55} / 2^{46.5} = 2^{8.5} = 362 \text{ seconds}
    - AES: 2^{127}/2^{46.5} = 2^{80.5} \text{ seconds, or } 2^{55} \text{ years}
    - DES “cracker”, built for $250K, broke DES in <3 days, in ’99

- Short block length
  - Repeated blocks happen “too frequently”

- Some (theoretical) attacks have been found
  - Claimed known to DES designers 15 years before public discovery!

- Non-public design process

Modes of encryption

- Used for encrypting a long message \( m_1, \ldots, m_n \)

- ECB
  - \( C_i = F_K(m_i); \) the ciphertext is \( (C_1, \ldots, C_n) \)
  - Should not be used
    - Why?
    - Not even secure against ciphertext-only attacks

- Cipher block chaining (CBC) –
  - single random block created, prepended to message
  - ciphertext \( c_i \) xor’d w/ \( p_{i+1} \)
  - IV; \( C_i = F_K(m_i \oplus C_{i-1}); \) the ciphertext is \( (IV, C_1, \ldots, C_n) \)
The effect of ECB mode

original

encrypted using ECB mode

*Images from Wikipedia

Modes of encryption

- **OFB (stream cipher mode)**
  - $z_i = F_k(z_{i-1}); C_i = z_i \oplus m_i$; the ciphertext is $(IV, C_1, ..., C_n)$
  - pad can be generated in advance (before msg create or arrive)
  - can be en/de-encrypted a bit at a time
  - if plain/cipher pairs known to Trudy, she can change msg by XOR' ing ciphertext w/ plaintext, and then w/ her message
  - inserted or deleted blocks in ciphertext
  - de-synchronized
  - garbled ciphertext bits
  - only garble the corresponding plaintext

- **CTR (stream cipher mode)**
  - $z_i = F_k(IV+i); C_i = z_i \oplus m_i$; the ciphertext is $(IV, C_1, ..., C_n)$
  - Random access during encryption or decryption
Other modes

- CBC, OFB, and CTR modes are secure against chosen-plaintext attacks

Message integrity

- Encryption does generally do not provide integrity
  - hash (msg digest)
  - MAC(msg authentication code)
    - useful when don’t need privacy, just integrity.
      - hash(<secret> | msg), or
      - run CBC, but just send the last block of the ciphertext (the “CBC residue”) w/ the plaintext.
MAC usage

- Shared key $k$
- Sender computes a tag $t$ on the message $m$ using $k$
- Receiver verifies the message/tag pair using $k$