# Encryption

Basic idea: store and transmit information in an encoded form that bears no obvious relationship with the original message.

Two principles:

Encrypted data must contain

- redundancy (detect fake messages)
- unique identifier (detect replayed messages)

## Secret-key Cryptography

- M: plain-text messages
- E: encrypted message
- K: encryption key
- $E = f(M, K)$ (f is reasonably efficient to compute)
- $M = f^{-1}(E, K)$ ($f^{-1}$ is reasonably easy to compute)
- $M = f'(E)$ (very hard to compute)
- $K = f''(M, E)$ (hard to compute)

- privacy: have to know the secret key to read the message
- authenticity: if it can be decrypted with the secret key, then the sender must have known the secret key

Problem: a secure channel is needed to distribute secret keys

## Public-key Cryptography

- M: plain-text messages
- E: encrypted message
- $K_1, K_2$: encryption key pair
- $E = f(M, K_1)$
- $M = f(E, K_2)$
- $M = f(f(M, K_1), K_2)$
- $M = f(f(M, K_2), K_1)$

Idea: publish one of the keys of a pair, keep the other key secret

- anyone can encrypt a message for you using your public key, only you can decrypt it using the secret key.
- if you send a message encrypted with your secret key, anyone can verify the authenticity of the message by decrypting it with your public key.

## Cryptographic Checksum (Message Digest)

- M: plain-text messages
- d: message digest (cryptographic checksum)
- $d = f(M)$ (very hard to invert f)

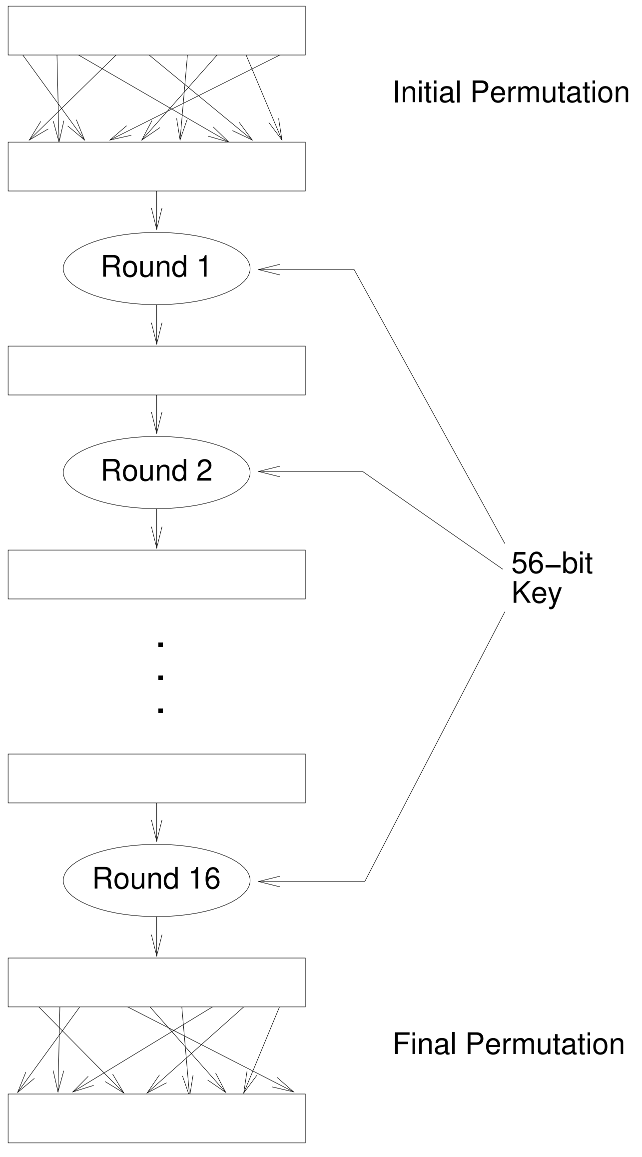
Can be combined with public-key encryption as an efficient method for ensuring message integrity:

- e.g., *keyed MD5*
- sender draws random number $k$
- sender transmits $m + MD5(m + k) + RSA(k, senderPrivateKey)$
- $m$: message
- $MD5(m + k)$: MD5 crypto checksum of message concatenated with random key
- $RSA(k, senderPrivateKey)$: RSA encrypted key, using sender's private key

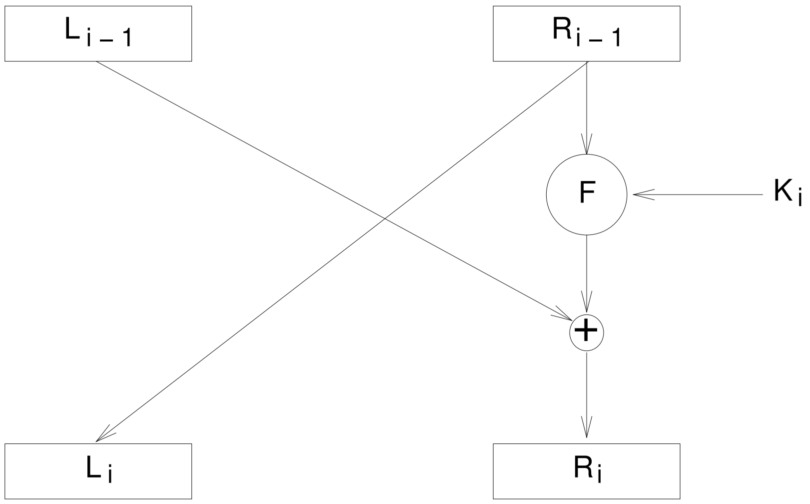
# Data Encryption Standard (DES)

A secret key encryption algorithm

- works on 64 bit blocks
- key size 56 bits



## Data Encryption Standard (DES)

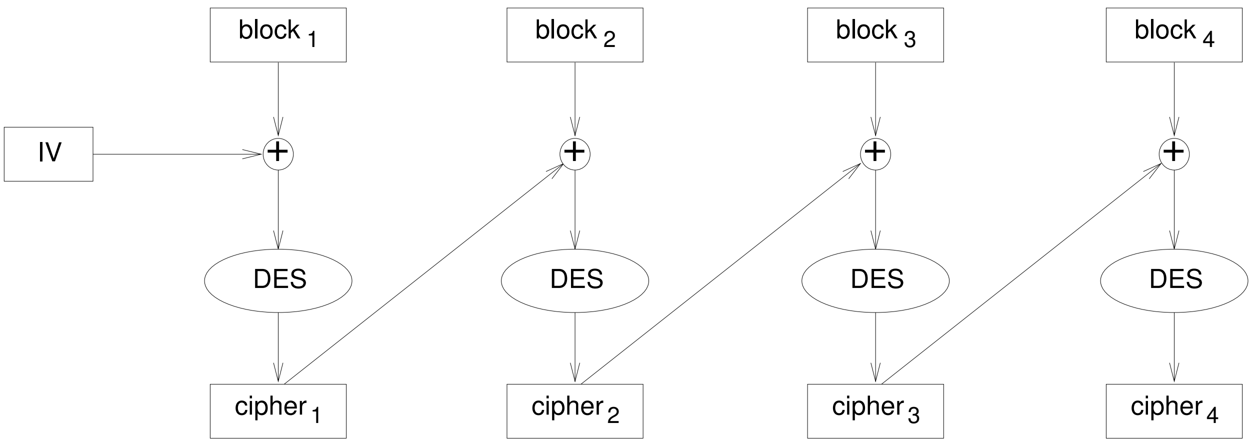



- $K_i$ at round $i$ derived from key through shifts and permutations according to a table
- combiner function $F$:
  - expand $R_{i-1}$ to 48 bits (see below)
  - XOR with $K_i$
  - reduce to 32 bits by compressing contiguous 6-bit chunks into 4 bits using an *S-box*.
  - S-box is a chunk-specific translation table

Decryption function is identical, except that $K_i$s are applied in reverse order.

## Cyclic Block Chaining

allows encryption of messages larger than 64 bits




IV = random number that is encrypted and transmitted with the message

## RSA Public Key Encryption

Key length is 512 bits

Choosing a pair of keys:

- pick two prime numbers $p, q$ (about 256 bits long)
- $n = p * q$
- choose $e$, such that $e$ and $(p-1)*(q-1)$ have no common factors greater than 1
- $d = \frac{mod((p-1)*(q-1))}{e}$
- key 1: $< e, n >$
- key 2: $< d, n >$
- discard $p, q$ (cannot be compromised)

Encryption: $E = M^e(mod(n))$

Decryption: $M = E^d(mod(n))$

$M$ must be no more than 512 bits long

longer messages encrypted by concatenation

# Performance

DEC 3000/600 (175Mhz Alpha)

- DES: 36 Mbps
- RSA: 1 Kbps
- MD5: 80 Mbps

Custom VLSI:

- DES: 1 Gbps
- RSA: 64 Kbps