CS530
Cryptography

Bill Cheng

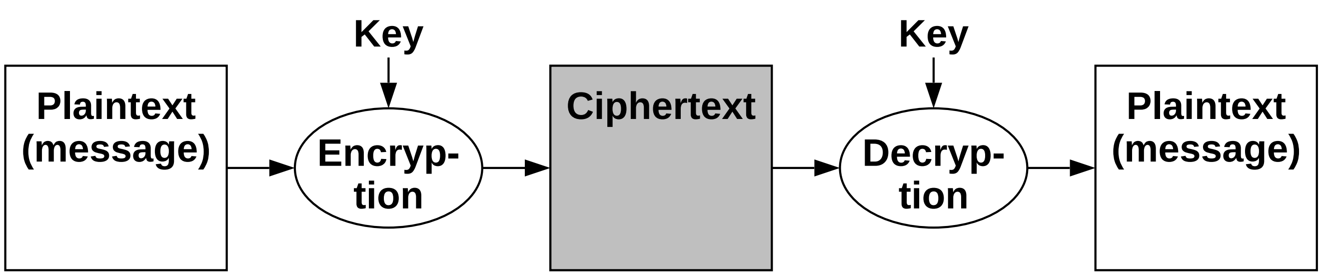
http://merlot.usc.edu/cs530-s10

Cryptography & Security

- Cryptography underlies many fundamental services
  - Confidentiality
  - Data integrity
  - Authentication
- Cryptography is *the* basic foundation of much of security

A Brief History

- *Steganography:* "covered writing"
  - Demaratus (5th century B.C.)
    - writing under wax on tablets
  - German microdots (WWII)
  - crucial flaw: Discovery yields knowledge
    - confidentiality through obscurity
  - covert channels
    - Ex: timing channel
- *Cryptography:* "secret writing"
  - TASOIINRNPSTO and TVCTUJUVUJPO



A Brief History (Cont...)

- Two basic types of cryptography
  - *Transposition* (TASOIINRNPSTO) or permutation
    - message broken up into units
    - units permuted in a seemingly random but reversible manner
    - Ex: wrap tape on rod
    - difficult to make it easily reversible only by intended receiver
    - exhibits same first-order (or mono-gram) statistics (but distort di-grams, tri-grams, etc.)
  - *Substitution* (TVCTUJUVUJPO)
    - (cont...)

A Brief History (Cont...)

- Substitution (TVCTUJUVUJPO)
  - message broken up into units
  - units mapped into ciphertext
  - Ex: Caesar cipher
  - first-order statistics are isomorphic in simplest cases
    - note: for transposition, first-order statistics are identical
  - predominant form of encryption

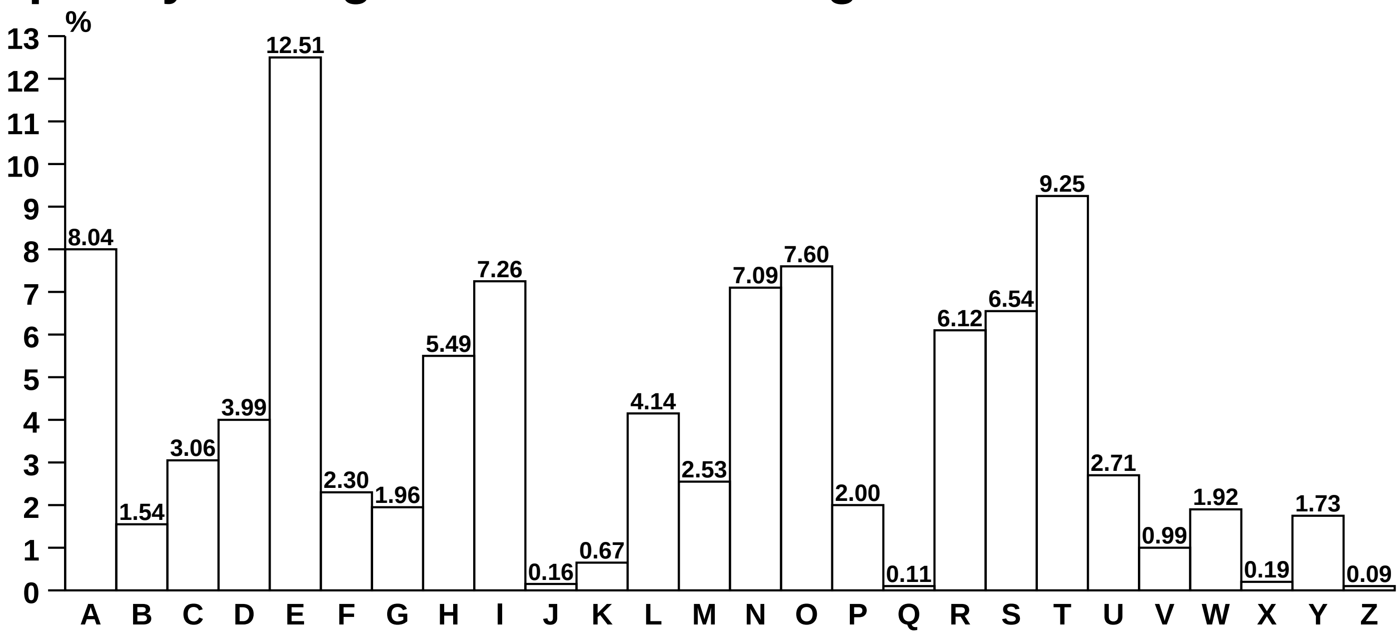
How Much Security?

- *Monoalphabetic substitution cipher*
  - permutation on message units: letters
  - 26! different permutations
  - each permutation considered a *key*
  - key space contains 26! = 4×10^{26} keys
    - equal to number of atoms in a gallon of water
    - equivalent to a 88-bit key (more than DES!)

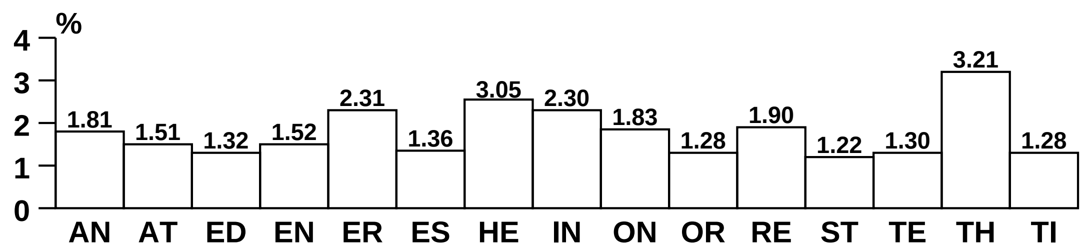
- So why not use substitution ciphers?
  - hard to remember 26-letter keys
    - but we can restrict ourselves to shorter keys
    - Ex: JULISCAERBDFGHKM, etc.
  - remember: first-order statistics are isomorphic
    - vulnerable to simple cryptanalysis

1964 English Language Statistics

- frequency of single characters in English text



- frequency of 15 common digrams in English text (27% overall)



Substitution Ciphers

- Two basic types
  - *symmetric-key* or conventional
    - single key used for both encryption and decryption
    - keys are typically short, because key space is densely filled
    - Ex: DES, 3DES, RC4, Blowfish, IDEA, etc
  - *public-key* or asymmetric
    - two keys: one for encryption, one for decryption
    - keys are typically long, because key space is sparsely filled
    - Ex: RSA, El Gamal, DSA, etc

Conventional Cryptography

- Stream cipher
  - ***stream cipher:*** **generates a (random or pseudorandom) keystream and applies it to a stream of plaintext with XOR**
    - **good for applications such as telnet**
    - **Ex: RC4**
  - ***one-time pad:*** **if the keystream is truely randomly chosen and never used again, the stream cipher is a one-time pad**
    - **the one-time pad can be shown to be *theoretically unbreakable***

RC4

```
/* state information */
static uns8 state[256], x, y;

void rc4init(uns8 *key,
        uns16 length)
    /* initialization */
{
    int i;
    uns8 t, j, k=0;

    for (i=256; i--; ) state[i] = i;

    for (i=0, j=0;
            i < 256;
            i++, j=(j+1)%length) {
        t = state[i];
        state[i] =
            state[k+= key[j] + t];
        state[k] = t;
    }
    x = y = 0;
}
```

```
uns8 rc4step()
    /*
     * return next
     * pseudo-random
     * octet
     */
{
    uns8 t;

    t = state[y += state[++x]];
    state[y] = state[x];
    state[x] = t;

    return state[
        state[x]+state[y]
    ];
}
```

RC4 (Cont...)

- To generate a random byte, do:

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

- Key scheduling algorithm:

```
for i from 0 to 255
    S[i] := i
j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod l]) mod 256
    swap(S[i],S[j])
```

Conventional Cryptography (Cont...)

- Block ciphers encrypt message in units called blocks
  - DES: 8-byte key (56 key bits), 8-byte block
    - $2^{56} \approx 10^{17}$

      Note: $2^{56} = 10^{X}$

      $56 \log 2 = X \log 10$

      $X = 56 \log 2 / \log 10$
  - larger blocks make simple cryptanalysis useless (at least for short messages)
    - not enough samples for valid statistics
    - "octo-gram statistics needed"

Key and Block Size

- Do larger keys make sense for an 8-byte block?
  - 3DES: Key is 112 or 168 bits, but block is still 8 bytes long (64 bits)
  - key space is larger than block space
  - *Q*: how many possible keys are out there?
  - *A*: equal to the size of the *permultation space*
  - why?
    - each key can be think of as a way to map an input pattern to an output pattern
    - *Q*: how many different patterns are there?
    - *A*: 2^{64}
    - remember, must be *one-to-one* mapping
  - but how large is permutation space?
    - $2^{64}! = ?$
    - use Stirling's Formula: $n! \approx n^n e^{-n} \sqrt{2\pi n}$

CS530
Cryptanalysis

Bill Cheng

http://merlot.usc.edu/cs530-s10

Cryptanalysis

- ***Cryptanalysis*** **is the study of mathematical techniques for attempting to defeat cryptographic techniques and information security services**
  - **a** ***cryptanalyst*** **is someone who engages in cryptanalysis**
- ***Cryptology*** **is the study of cryptography and cryptanalysis**
- **Six general types of cryptanalytic attacks:**
  - ***ciphertext-only attack***
  - ***knwon-plaintext attack***
  - ***chosen-plaintext attack***
  - ***adaptive-chosen-plaintext attack***
  - ***chosen-ciphertext attack***
  - ***adaptive-chosen-ciphertext attack***
- **Another type of cryptanalytic attack**
  - ***purchase-key attack***

Cryptanalytic Attacks

- ***Ciphertext-only attack***
  - given ciphertexts
  - deduce plaintexts (or key)
- ***Known-plaintext attack***
  - given plaintext-ciphertext pairs
  - deduce key
- ***Chosen-plaintext attack***
  - the cryptanalyst has access to the *encryption* device
  - given plaintext→ciphertext pairs of the attacker's choosing
  - deduce key
- ***Adaptive-chosen-plaintext attack***
  - special case of a chosen-plaintext attack
  - the cryptanalyst can modify his choice based on the result of previous encryption

Cryptanalytic Attacks (Cont...)

- *Chosen-ciphertext attack*
  - the cryptanalyst has access to the *decryption* device
  - given ciphertext$\rightarrow$plaintext pairs of the attacker's choosing
  - deduce key
- *Adaptive-chosen-ciphertext attack*
  - special case of a chosen-ciphertext attack
  - the cryptanalyst can modify his choice based on the result of previous decryption
- *Rubber-hose cryptanalysis (*or *purchase-key attack)*
  - the cryptanalyst threatens, blackmails, or tortures someone until they give him the key
  - often the best way to break an algorithm!

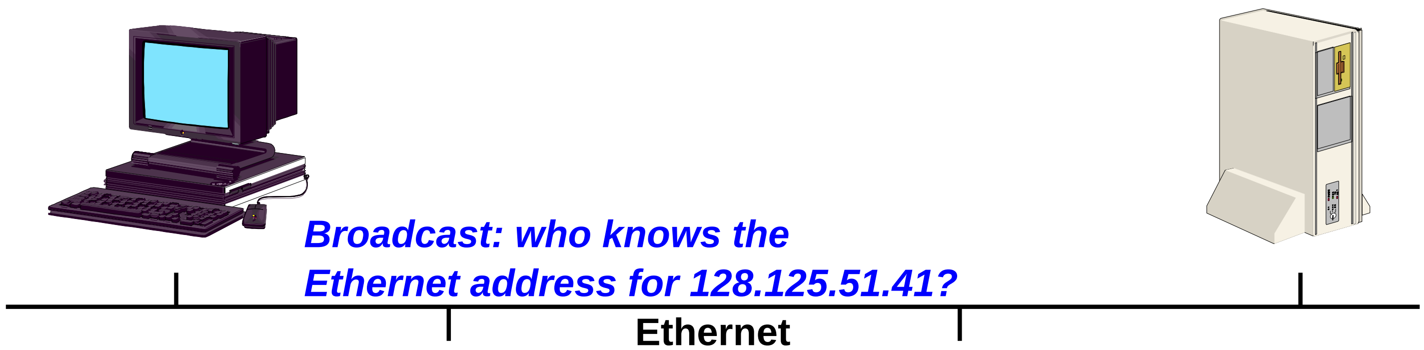
Attack on Protocols

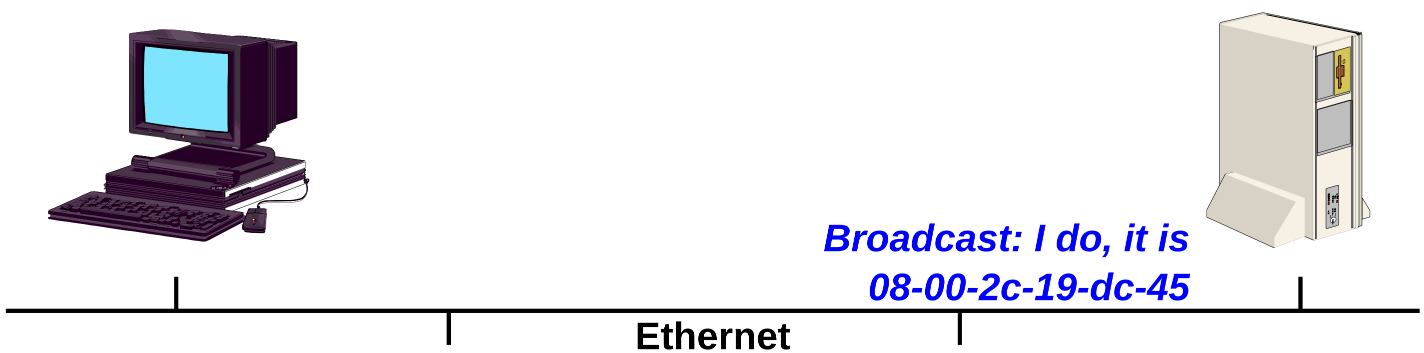
- Until a protocol is proven to provide the service intended, the list of possible attacks can never be said to be complete
  - *known-key attack:* an adversary obtains some keys used previously to determine new keys
  - *replay attack:* an adversary records a communication session and replays the entire session, or a portion thereof, at some later point in time
  - *impersonation attack:* an adversary assumes the identity of one of the legitimate parties in a network
  - *dictionary attack:*
    - usually an attack against passwords
    - password is stored as image of unkeyed hash function
    - an adversary takes a list of possible passwords, hash all entries and compare with stored hash values

Attack on Protocols (Cont...)

- ***forward search attack:*** **similar to the dictionary attack and is used to decrypt messages**
- ***interleaving attack:*** **involves some form of impersonation in an authentication protocol**
  - **A and B executes a security protocol**
  - **an adversary intercepts all messages and sends its own messages**
    - **man-in-the-middle attack**

Finding Ethernet Address: Address Resolution (ARP)





Man-in-the-middle Attack

- ARP weaknesses
  - accept additional ARP responses even if an ARP response has already been received
    - overwrite cached ARP value
  - accept ARP response even if no ARP request
  - fixes for this exist, but often not implemented or installed
- Man-in-the-middle attack
  - an attacker on LAN and send an ARP response to a host H to impersonate the gateway G
    - and tells G that its ethernet address is that of H
  - now the attacker can intercept and modify any packet that goes between H and G

CS530
DES

Bill Cheng

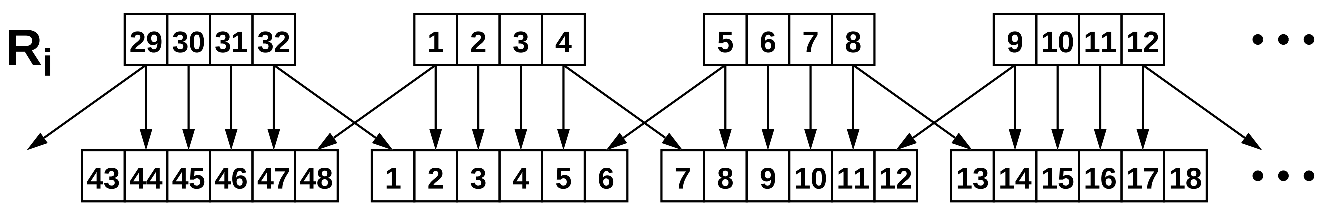
http://merlot.usc.edu/cs530-s10

Anatomy Of A Block Cipher

- DES: Data Encryption Standard
  - developed as Lucifer (one of a few) at IBM in 1970s
  - break message into 8-byte (64-bit) blocks
    - each block broken into 32-bit halves
    - initial permutation
    - 16 rounds of scrambling
    - final (reverse) permutation
  - Feistel Network structure

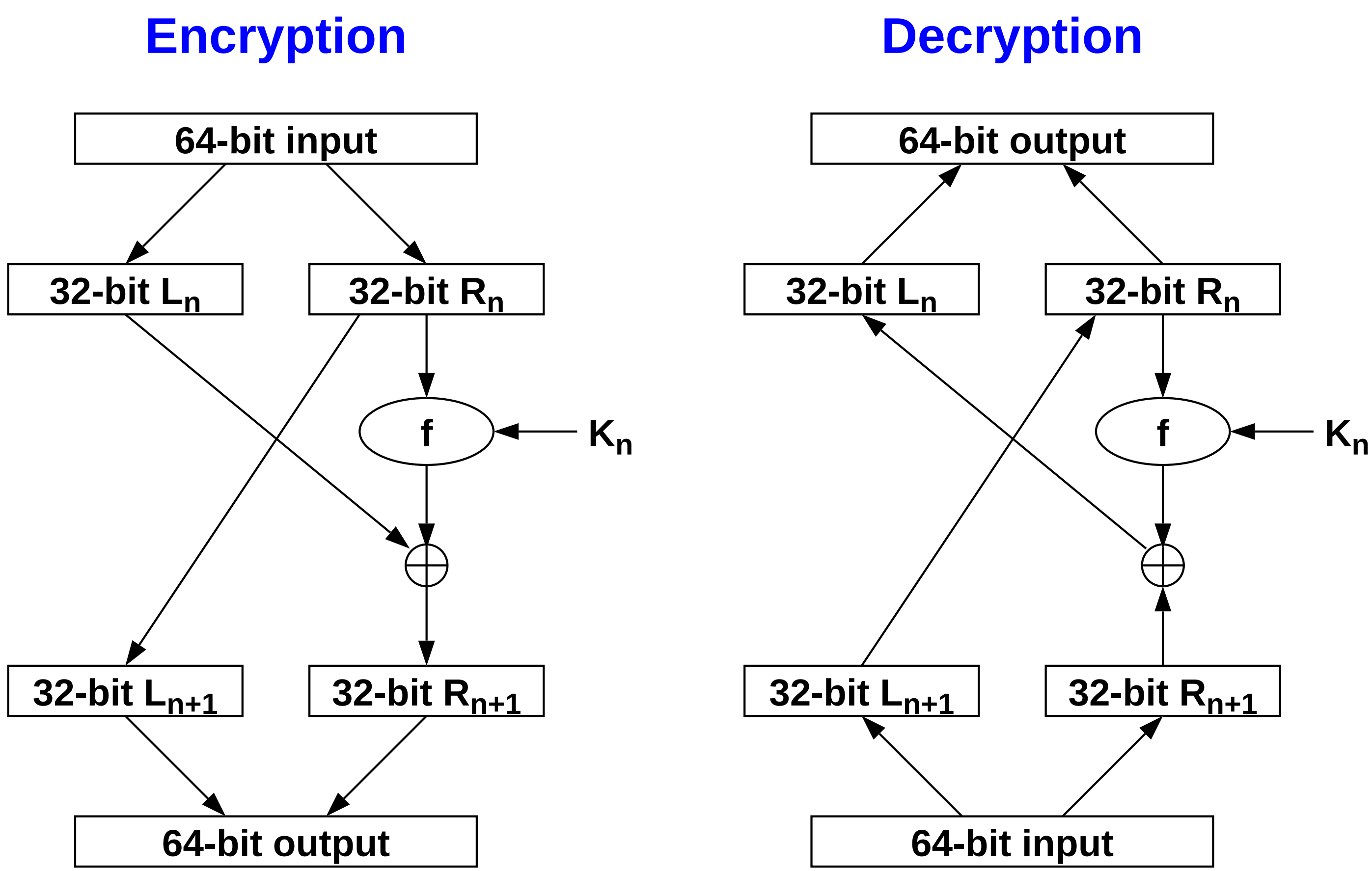
The Scrambling Function

- In each round i, we have L_i and R_i
  - $L_{i+1} = R_i \leftarrow$ typical of Feistel networks
  - $R_{i+1} = L_i \oplus f(R_i)$
- f-function
  - key is compressed and permuted to 48 bits (called subkeys, different for each round)
  - R_i (32-bits) is expanded and permuted to 48 bits



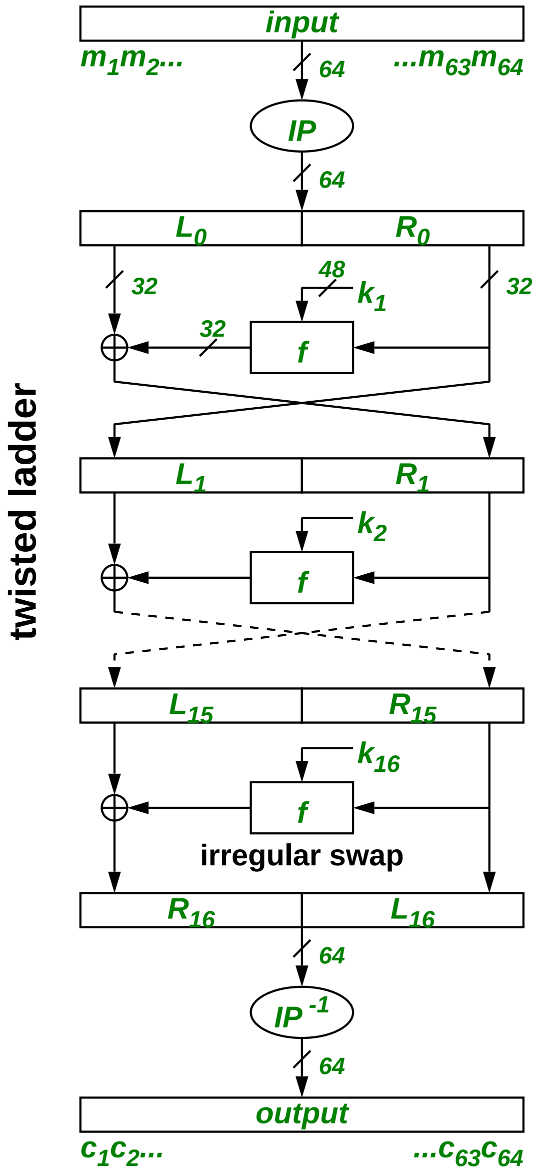
  - 48 bits XOR'd, passed through S-boxes (to produce 32 bits), then permuted again
  - irreversible

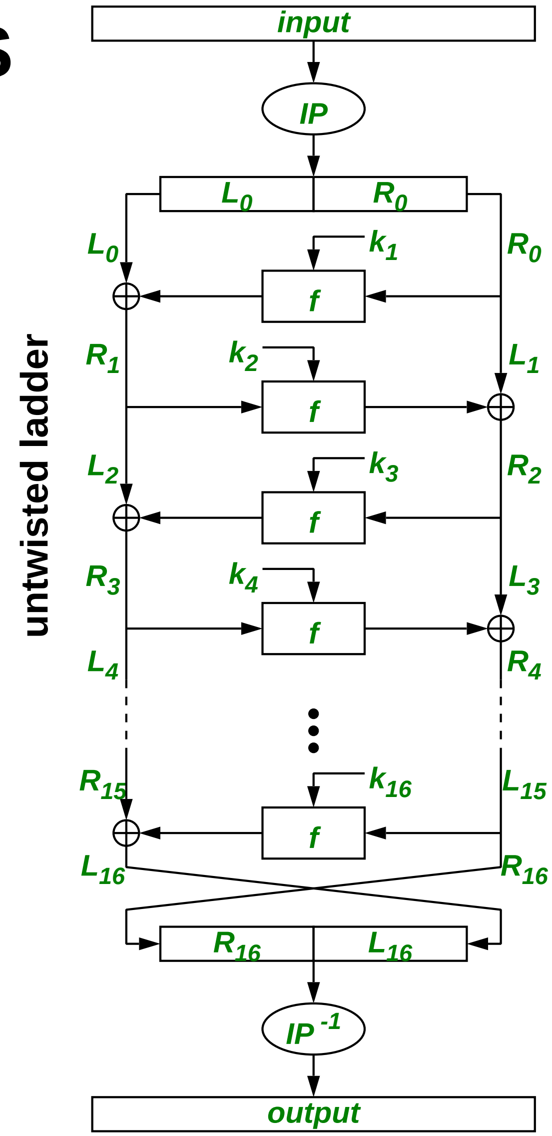
Feistel Network



- f only used in only one direction for both encryption and decryption

DES

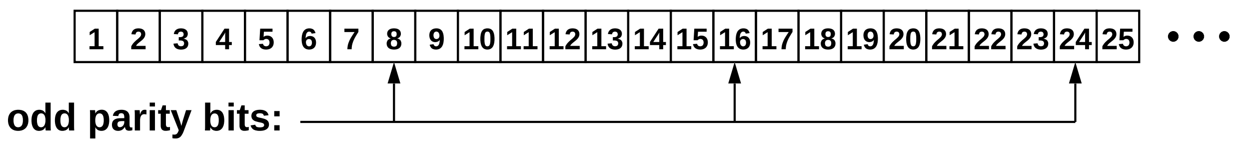

input
m_1m_2...
...$m_{63}m_{64}$
64
IP
64
L_0
R_0
32
48
k_1
32
32
f
twisted ladder
L_1
R_1
k_2
f
L_{15}
R_{15}
k_{16}
f
irregular swap
R_{16}
L_{16}
64
IP^{-1}
64
output
c_1c_2...
...$c_{63}c_{64}$


input
IP
L_0
R_0
k_1
f
R_1
L_1
k_2
L_2
R_2
k_3
R_3
L_3
k_4
L_4
R_4
untwisted ladder
R_{15}
L_{15}
k_{16}
L_{16}
R_{16}
IP^{-1}
output

Key Compression

- **Reduction to 56 bits (no parity bits)**

  DES key (64 bits);



- **Broken into halves**
  - **each half is rotated by 1 or 2 bits**
  - **48 bits out of 56 selected**
- **Why do this?**
  - **use a different set of bits for each round**
  - **not exactly symmetric**

Data Expansion

- Data broken into 4-bit groups
- Each group expanded to 6 bits
- Why do this?
  - match subkey length
  - data diffusion occurs faster

Substitution Boxes (S-Boxes)

- This is the heart of DES
- 48 bit result broken into 6-bit units
- Each unit passed through an S-box
  - 6-bit input, 4-bit output
  - each S-box is a 4x16 array of 4-bit numbers
  - b1 and b6 specify row, b2 through b5 specify column
- End result passed through P-box

DES Properties

- Desirable characteristics for a block cipher
  - each bit of the ciphertext should depend on all bits of the key and all bits of the plaintext
  - there should be no statistical relationship evident between plaintext and ciphertext
  - altering any single plaintext or key bit should alter each ciphertext bit with probability *1/2*
  - altering a ciphertext bit should result in unpredictable change to the recovered plaintext block
- Empirically, DES satisfies all the above objectives

DES Weak Keys

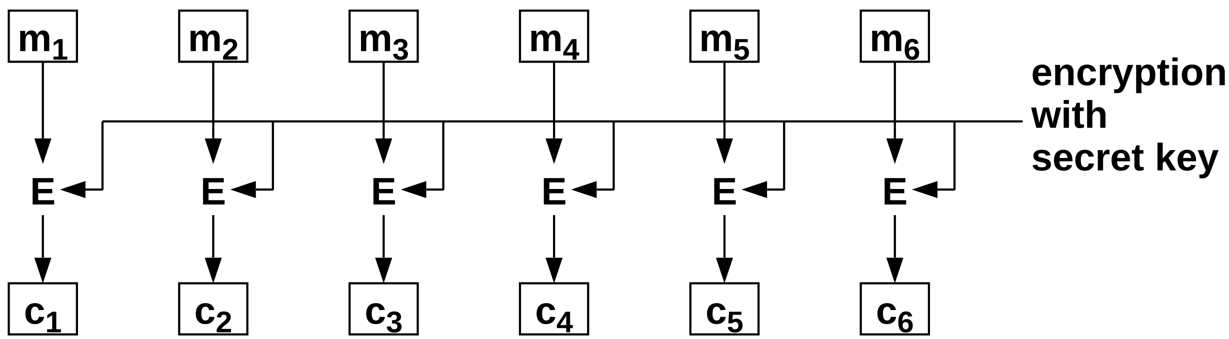
- If generated *subkeys* are such that $k_1 = k_{16}$, $k_2 = k_{15}$, and so on, the encryption and decryption functions coincide
  - these are called ***weak keys*** (and also ***palindromic keys***)
    - if K is a weak key, then $E_K(E_K(x)) = x$ for all x
- DES also has ***semi-weak keys***
  - if (K_1,K_2) is pair of semi-weak keys, then $E_{K_1}(E_{K_2}(x)) = x$ for all x
- DES has 4 weak keys and 6 pairs of semi-weak keys

semi-weak keys (hexadecimal)
01FE 01FE 01FE 01FE, FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1, E01F E01F F10E F10E
01E0 01E0 01F1 01F1, E001 E001 F101 F101
1FFE 1FFE 0EFE 0EFE, EF1F EF1F EF0E EF0E
011F 011F 010E 010E, 1F01 1F01 0E01 0E01
E0FE E0FE F1FE F1FE, FEE0 FEE0 FEF1 FEF1

weak keys (hexadecimal)
0101 0101 0101 0101
FEFE FEFE FEFE FEFE
1F1F 1F1F 1F1F 1F1F
E0E0 E0E0 E0E0 E0E0

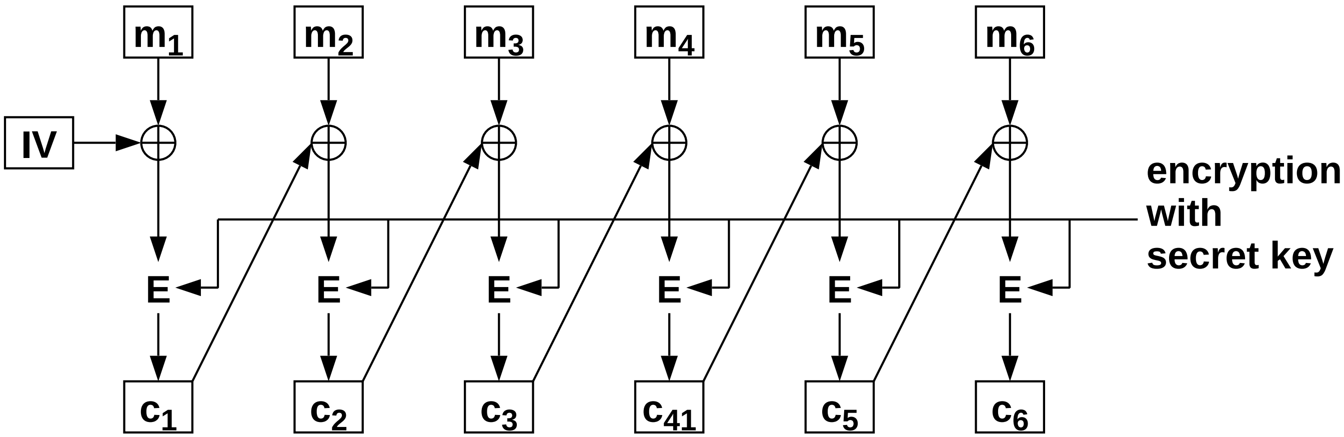
Modes of DES Operation

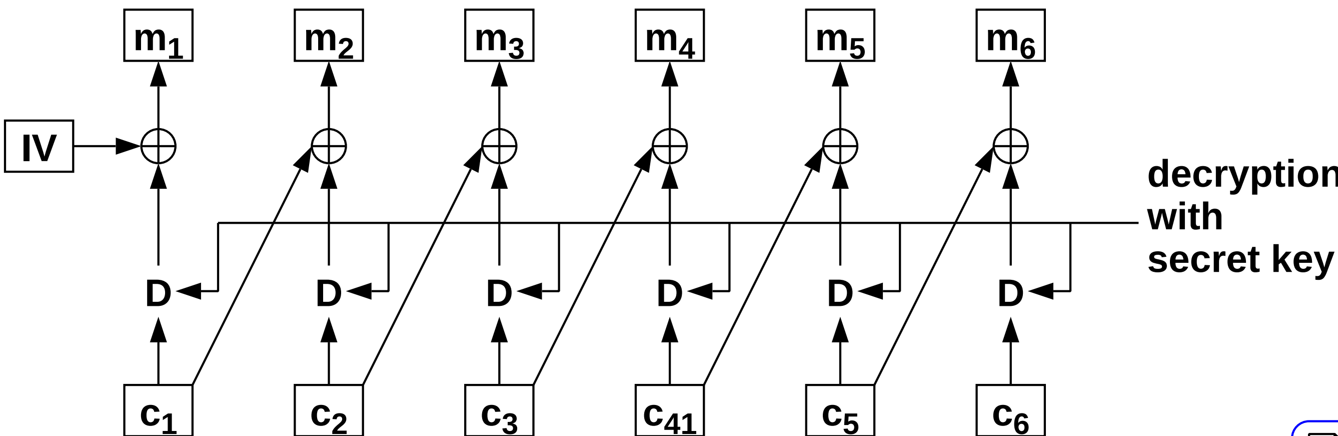
- What to do if message is longer than 8 bytes?
- Electronic Codebook (ECB)
  - each block encrypted in isolation
  - vulnerable to block replay (same input ⇒ same output)



- Cipher Block Chaining (CBC)
  - each plaintext block XOR'd with previous ciphertext before encryption
  - easily incorporated into decryption
  - what if prefix is always the same? IV!

Cipher Block Chaining (CBC)



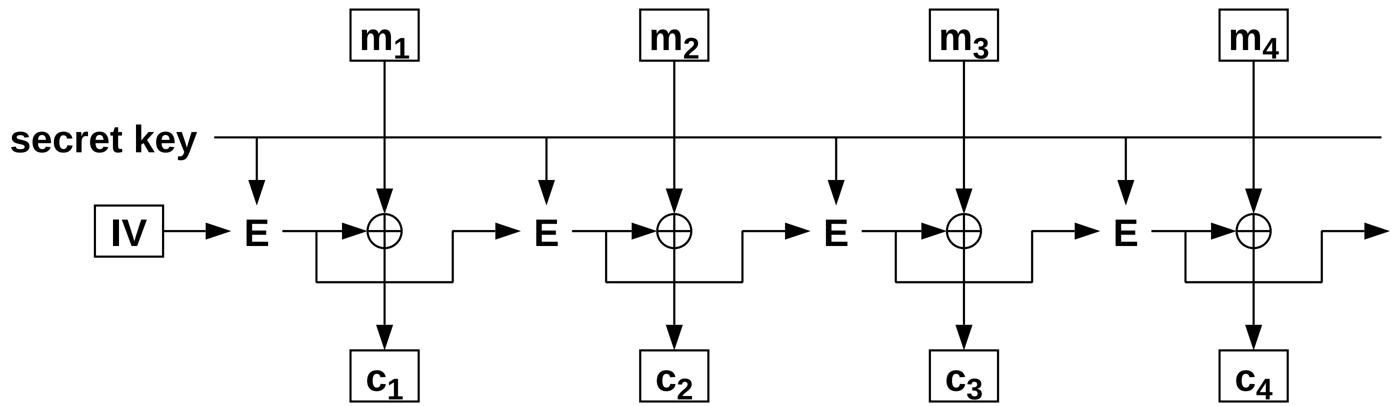


Modes of DES Operation (Cont...)

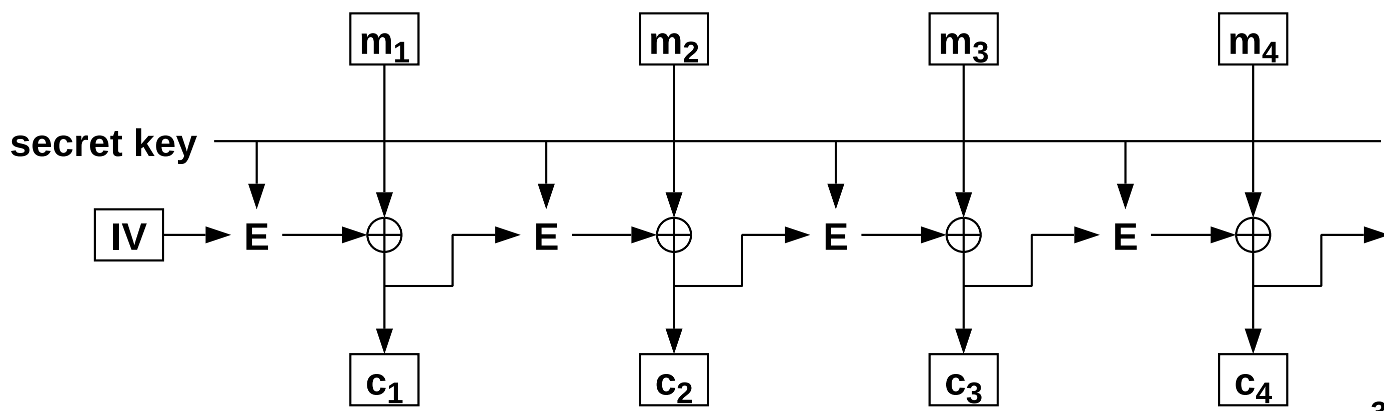
- Stream cipher
  - *stream cipher:* generates a (random or pseudorandom) keystream and applies it to a stream of plaintext with XOR
  - *one-time pad:* if the keystream is truely randomly chosen and never used again, the stream cipher is a one-time pad
- Cipher Feedback (CFB)
  - for encrypting character-at-a-time (or less)
  - chains as in CBC
  - also needs an IV
    - must be unique
- Output Feedback (OFB)
  - like CFB, but some bits of output fed back into input stream

OFB vs CFB

- OFB (simplified): $v_i = E(k, v_{i-1})$ and $c_i = m_i \oplus v_i$

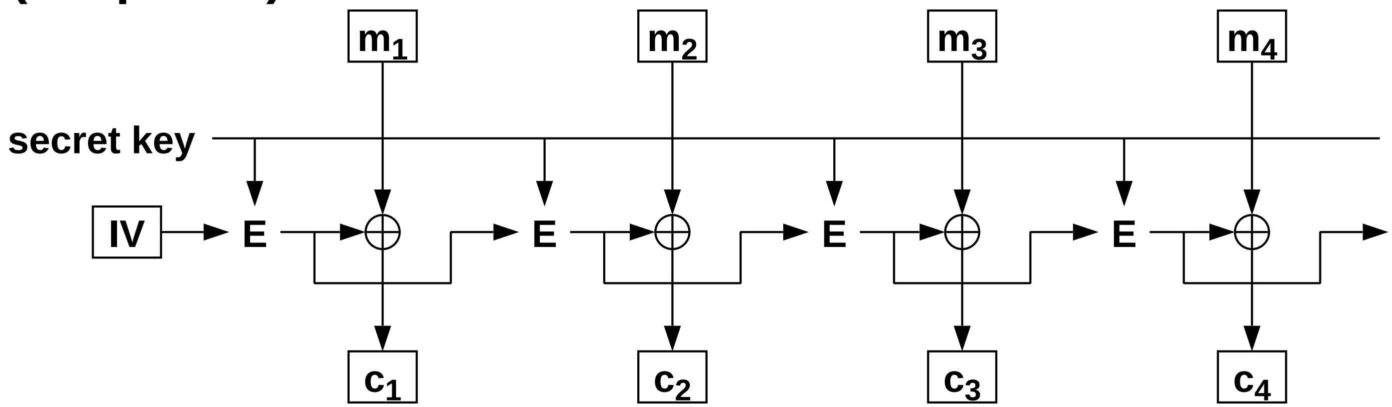


- CFB (simplified): $c_i = m_i \oplus E(k, c_{i-1})$

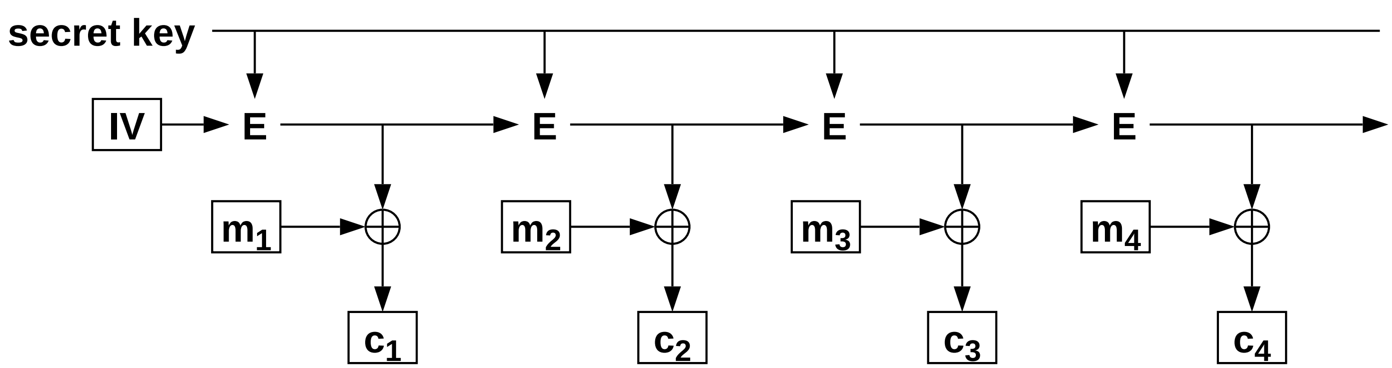


OFB vs CFB (Cont...)

OFB (simplified)



Alternate view:



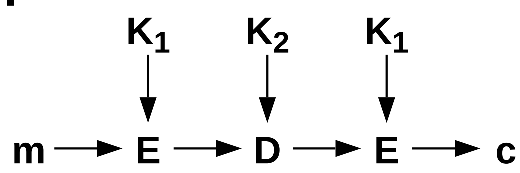
DES Variants and Applications

- Crypt: Unix hash function for passwords
  - uses variable expansion permutations
  - add a 12-bit *salt* (to modify DES)
    - to mitigate the ***precomputed dictionary attack***
  - encrypt the number 0
- DES with key-dependent S-boxes
  - cannot be done blindly

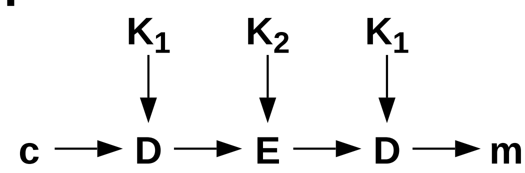
Variants and Applications (Cont...)

- **3DES: Encrypt using DES 3x**
  - **two and three-key types**

encryption:

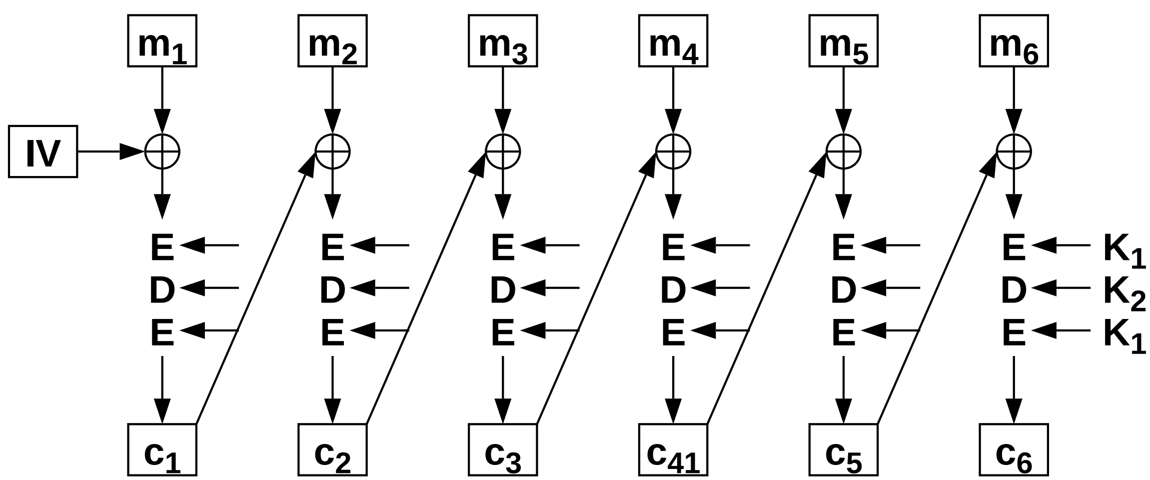


decryption:



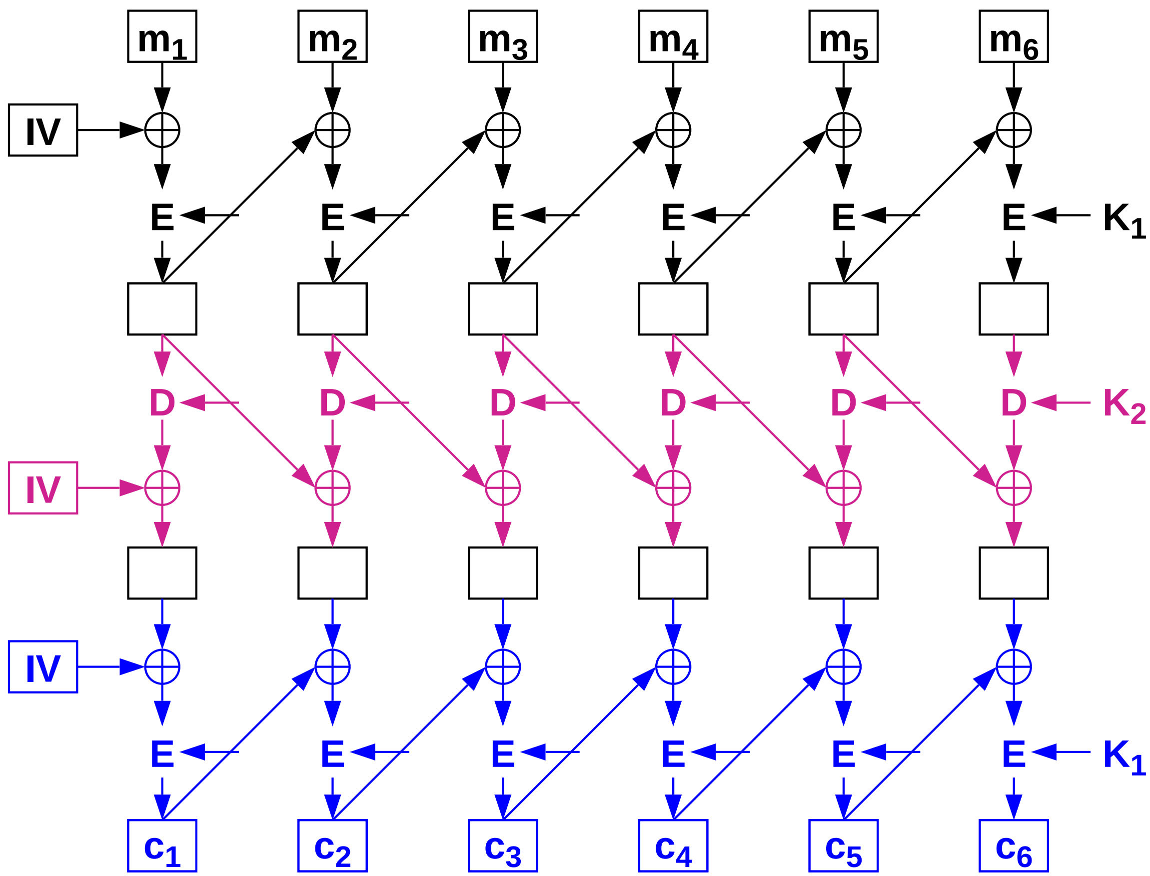
  - **inner and outer-CBC modes**

CBC on the outside:



Variants and Applications (Cont...)

- inner and outer-CBC modes (cont...)
  - CBC on the inside:

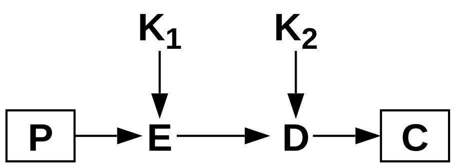


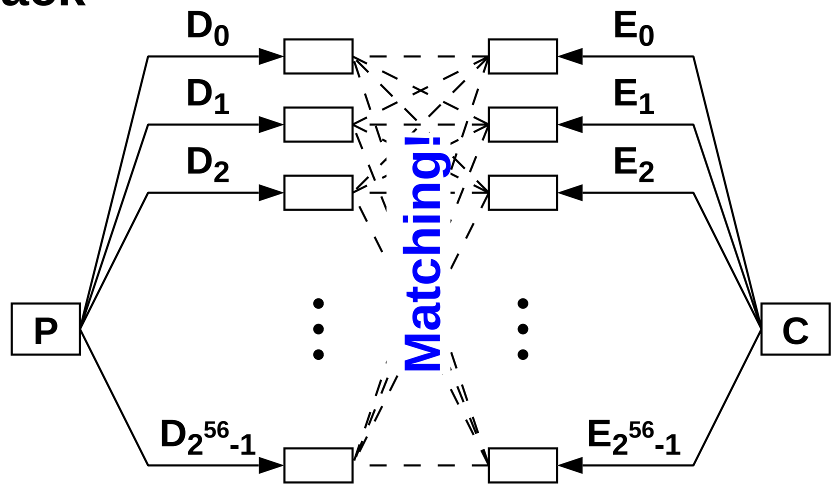
Variants and Applications (Cont...)

- 3DES (cont...)
  - inner-CBC mode for 3DES is more efficient, but less secure
    - more efficient because of possible pipelining
    - under some attacks inner-CBC mode is significantly weaker than outer-CBC mode; against other attacks based on block size, inner-CBC mode appears stronger (please note that this is different from what the textbook says)
  - main reason for EDE is backwards compatibility with single-key DES
- Why not 2DES? (EE, DE, or ED)
  - (cont...)

Variants and Applications (Cont...)

- **Why not 2DES? (EE, DE, or ED)**
  - **turns out 2DES is not much more secure than DES**
  - ***meet-in-the-middle* attack**





Attacks on DES

- No known systematic attack (for 16 rounds)
  - is DES "closed" (that is, a group)?
    - does there exist a K_3 such that $E_{K_1}(E_{K_2}(P)) = E_{K_3}(P)$
    - if it were, double encryption would be useless
    - DES is *not* closed
  - is DES "pure"?
    - does there exist a K_4 such that $E_{K_1}(E_{K_2}(E_{K_3}(P))) = E_{K_4}(P)$
    - if it were, triple encryption would be useless
    - unfortunately, don't know if DES is pure
  - does DES has a skeleton/allpass key?
    - not likely because DES is symmetric
- Brute force attacks only
  - try all 2^{56} keys!

Lucifer Goes Standard

- Lucifer is one of the IBM ciphers
  - generally regarded in 1970s as one of the strongest cryptosystems
- Heading toward standardization as DES
  - NSA managed to get key size reduced to 56 bits (from 64), yielding 10^{17} keys
  - also apparently changed S-boxes
  - why (or why not) do this?
    - NSA does not trust IBM
    - who trusts NSA?

Certification of DES

- Had to be recertified every ~5 years
  - 1983: Recertified routinely
  - 1987: Recertified after NSA tried to promote secret replacement algorithms
    - withdrawal would mean lack of protection
    - lots of systems then using DES
  - 1993: Recertified after continued lack of alternative

CS530
AES

Bill Cheng

http://merlot.usc.edu/cs530-s10

Enter AES

- 1998: NIST finally refuses to recertify DES
  - 1997: Call for candidates for Advanced Encryption Standard (AES)
  - fifteen candidates whittled down to five
  - criteria: Security, but also efficiency
    - compare Rijndael with Serpent (which is generally regarded as more secure but less efficient)
  - 2000: Rijndael selected as AES

Structure of Rijndael

- Unlike DES, operates on whole bytes for efficiency of software implementations
- Key sizes: 128/192/256 bits
- Variable rounds: 9/11/13 rounds
- Rounds are not Feistel networks
- Round structure
  - run block through S-box (8x32)
  - permute result into 4x4/4x6/4x8 array of bytes
  - multiply each byte by 1, 2, or 3 in GF(2^8)
    - addition in GF(2^8) is done through XOR
  - mix subkey into result

Security of Rijndael

- Based on arithmatic in GF(2^8)
- Key size is enough
- Immune to linear or differential analysis
- But Rijndael is a very structured cipher
  - S-box consists of byte reciprocals in GF(2^8)
    - finite field $Z_2[x] / (x^8 + x^4 + x^3 + x + 1)$
  - permutations are regular
- Attack on Rijndael's algebraic structure
  - breaking can be modeled as equations
    - only need to know a single plaintext/cipher text pair
    - ~8,000 quadratic equations with ~1,600 variables (also in GF(2^8))

Impact of Attacks on Rijndael

- Currently of theoretical interest only
  - reduces complexity of attack to about 2^{100} (after the system of quadratic equations are solved)
  - also applicable to Serpent (complexity is about 2^{200})
- Still, uncomfortably close to feasibility
  - DES is already insecure against brute force
  - Schneier (somewhat arbitrarily) sets limit at 2^{80}
- Certainly usable pending further results