

# Computer Security

## Secret Key Cryptography

# Secret Key Cryptography

- DES
- IDEA
- Skipjack
- AES

# Generic Block Encryption

- cryptographic algorithms take a fixed-length block and fixed length key
- generate an output block of same size as input

# Input Block Length

- remember keys must be long enough so that trying all keys is infeasible
- same reasoning applies to input block length
- too long a block length only effects performance

## Input Block Length

- DES, IDEA and Skipjack use 64 bits
- this means a useful number of plaintext, ciphertext pairs almost impossible to obtain and manage

5

## General Encryption

- take each  $2^{64}$  bit input value and map it to a unique one of the  $2^{64}$  bit output values
- would take  $\sim 2^{70}$  bits to specify this mapping
- this number is, in effect, a key
- not very practical

6

## Randomness

- secret key schemes take a reasonable key (eg., around 128 bits)
- used to generate a one-to-one mapping
- mapping, to someone who does not know key, should look random

7

## Bit Change

- any bit change in input should result in totally independent output
- eg., a change in the 3rd input bit should not always change the 12th output bit
- cryptographic algorithms should spread bits around
- if one input bit changes half the output should be the same and half should change

8

## Transformations

- simple transformations on blocks of data
  - substitutions
  - permutations

9

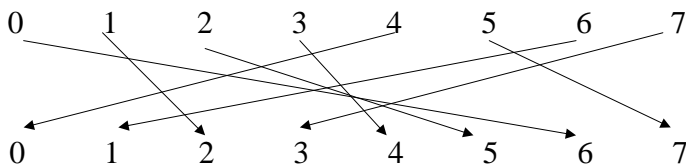
## Substitution

- for each of the  $2^k$  possible values of the input, the k-bit output is specified
- impractical for 64 bit input
- practical for 8 bit input

10

## Permutations

- specifies, for each of the k input bits, the output position to which it goes
- for k bit input takes  $k \log k$  bits
- Simple example



11

## Secret Key Algorithm

- break input into manageable size (eg., 8 bits)
- do a substitution on each small chunk
- take results and run them through a permuter
- repeat, so each bit is used as input to each substituter

12



## 56 bits

- 256 times easier to break than a real 64 bit key
- can parity check, but 1 in 256 chance of incorrectly accepting
- perhaps so the US govt can break it?

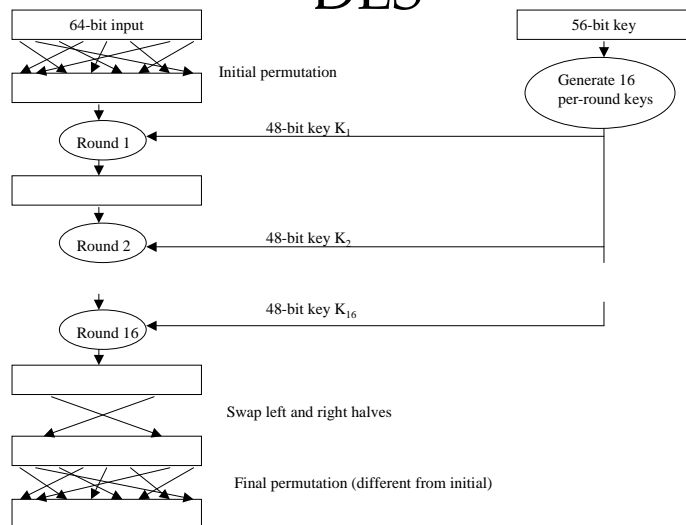
17

## DES Overview

- 64 input permuted to 64 bit result
- 56 bit key used to generate 16 48 bit per round keys
- each round takes as input the output of previous
- after 16th 64 bit output has its halves swapped
- then a final permutation

18

## DES



19

## Decryption

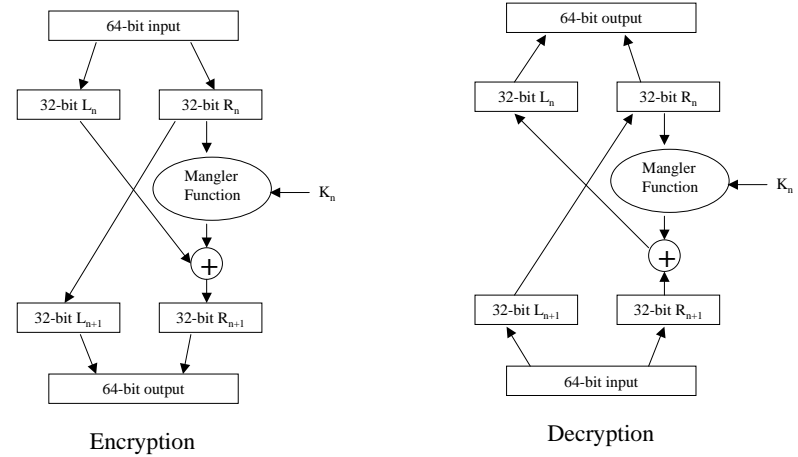
- run DES backwards
- initial and final permutations are inverses

20

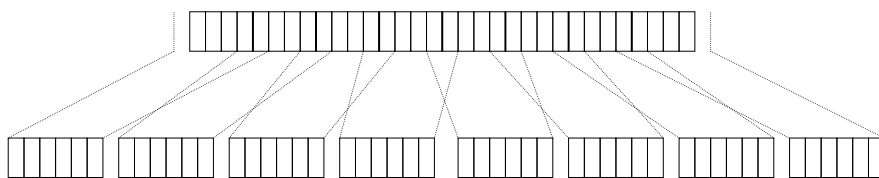
# Detail

- need to specify
  - permutations
  - how keys are generated
  - what happens in each round
- I expect you to look this up for yourself

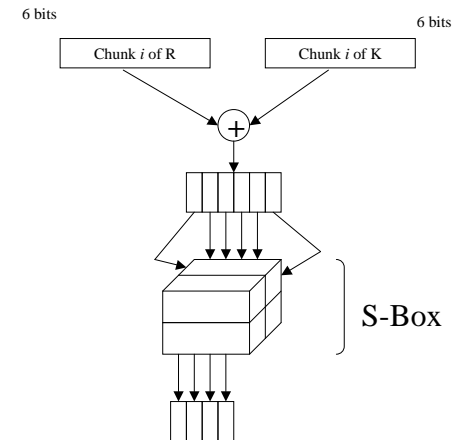
# DES Round



# Expansion of Rn from 32 to 48 bits



# S-Box



## Sample - S-Box 1

1&6	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1010	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

25

## Discussion

- is a secret key algorithm just shuffling some bits?
- DES is more subtle than that
- switching the order of S-boxes can lessen the security
- the design process was not made public
  - who knows exactly what cryptanalytic attacks it was designed to be proof against?

26

## IDEA

- International Data Encryption Algorithm
- efficient in software
- 64 bit block of input to 64 bit block of output
- 128 bit key

27

## IDEA and DES

- both operate in rounds
- both have a mangler function
  - both manglers do not have to be reversible
- DES uses same 48 bit keys for encryption and decryption
- in IDEA sub-keys related in more complex manner for encryption and decryption

28

## IDEA - Primitive Operation

- two 16 bit quantities mapped into a 16 bit quantity
- DES s-box maps 6 bits into 4
- IDEA uses three operations, all reversible
- important for decryption

29

## Operations

- bitwise exclusive or
- a slightly modified add
- a slightly modified multiply
- modified as result must be 16 bits

30

## Addition

- carries thrown away
- so addition mod  $2^{16}$

31

## Multiply

- calculate 32 bit result
- take remainder when divided by  $2^{16} + 1$
- this can be reversed

32

## Key Expansion

- 128 bit key expanded in 52 16-bit keys
- key expansion different for encryption and decryption
- once keys are generated encryption and decryption operations are the same

33

## Key Expansion - Encryption

- first 8 keys generated by cutting 16 bits at a time from the 128, from the left
- then do same again, starting at bit 25
- then again, offsetting by 25 more bits, and so on
- keys 50 and 51 are swapped

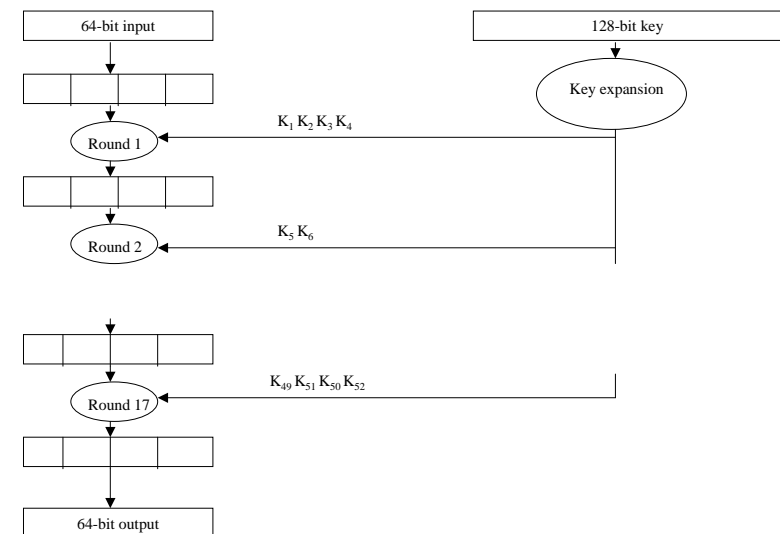
34

## IDEA Rounds

- 17 rounds
- odd rounds different from even rounds

35

## Basic IDEA Structure



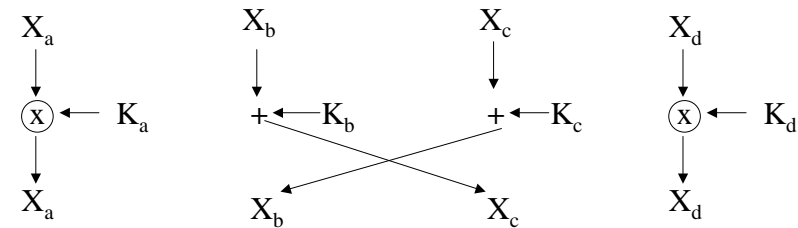
36

## One Round

- each round takes 64 bit input
- treats it as four 16 bit quantities ( $X_a, X_b, X_c, X_d$ )
- mathematical functions yield new versions of these four quantities
- odd rounds use four of the 16 bit keys
- even rounds use two of the 16 bit keys

37

## Odd Rounds



38

## Even Round

- more complicated than an odd round
- but easily reversible (see textbook for how)

39

## Even Round Modification Functions (Mangler)

$$Y_{in} = X_a \oplus X_b$$

$$Z_{in} = X_c \oplus X_d$$

$$Y_{out} = ((K_e \otimes Y_{in}) + Z_{in}) \otimes K_f$$

$$Z_{out} = (K_e \otimes Y_{in}) \oplus Y_{out}$$

We compute the new  $X_a$ ,  $X_b$ ,  $X_c$ , and  $X_d$

$$\text{New } X_a = X_a \oplus Y_{out}$$

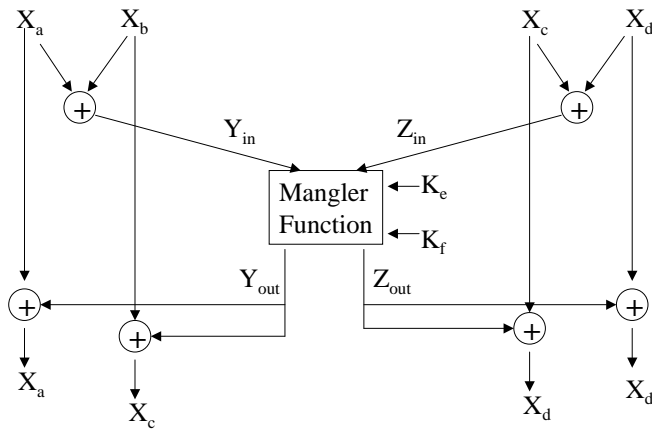
$$\text{New } X_b = X_b \oplus Y_{out}$$

$$\text{New } X_c = X_c \oplus Z_{out}$$

$$\text{New } X_d = X_d \oplus Z_{out}$$

40

## Even Rounds



41

## Decryption

- IDEA designed so same code can perform encryption or decryption given different expanded keys
- need inverses of the encryption keys (multiplicative inverse mod  $2^{16} + 1$ )
- use them in opposite order

42

## Discussion

- brute force on 128 bit key requires enormous computing resources
- nobody has published a way to break IDEA

43

## Skipjack

- Data Encryption Standard
- 64 bit input to 64 bit output
- 80 bit key
- developed by US govt National Security Agency in late 80's
- started in use in 1993
- declassified 1998

44

## Skipjack

- Used in the clipper chip and the fortezza PC card
- clipper chips can be used in telephones, faxes, modems
- US govt retains keys to decrypt all communication using this technology
- however, plan failed in face of widespread public opposition

45

## Skipjack

- Uses 32 rounds
- see additional material for details

46

## Use of Secret Key Cryptography

- we've looked at how to encrypt 64 bit chunks using secret key
- messages are often longer than that
- how are these algorithms used for large messages and integrity?

47

## Large Messages

- four schemes defined for DES
- can also be used for IDEA & Skipjack
- they are
  - electronic code book (ECB)
  - cipher block chaining (CBC)
  - k-bit output feedback mode (OFB)
  - k-bit cipher feedback mode (CFB)

48

## Electronic Code Book

- the obvious method
- encrypt each 64 bit block with secret key
- decrypt it at other end

49

## Problems

- two identical 64-bit blocks have identical ciphertext
- may be able to calculate positions of information fields

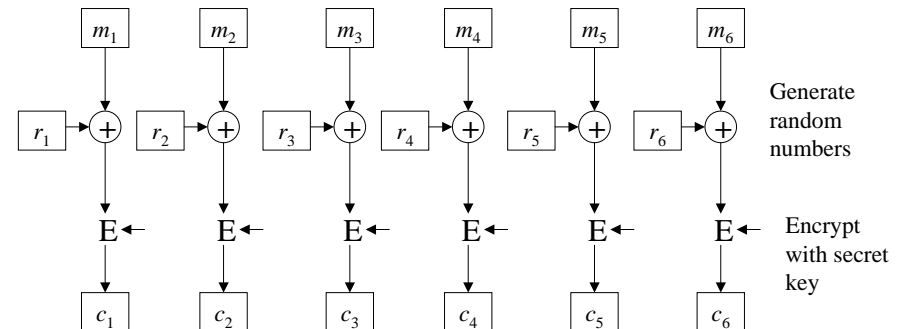
50

## Cipher Block Chaining

- with CBC identical plaintext does not lead to identical ciphertext
- first an example which isn't CBC but will help
- transmit both random r's and ciphertext

51

## Randomised ECB



52

## Problems

- twice as much information must be transmitted (r's as well as c's)
- attacker can still interfere with each individual block

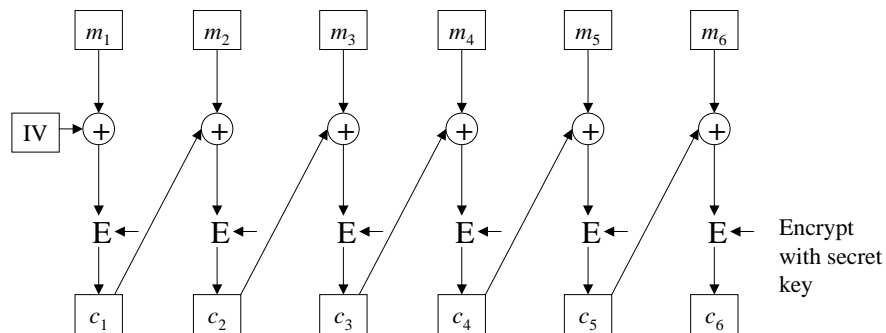
53

## CBC

- uses previous ciphertext as random number for next block encryption
- need a random number to start with
- known as IV (initialisation vector)
- randomly chosen IV's protect identical messages
- IV must be transmitted

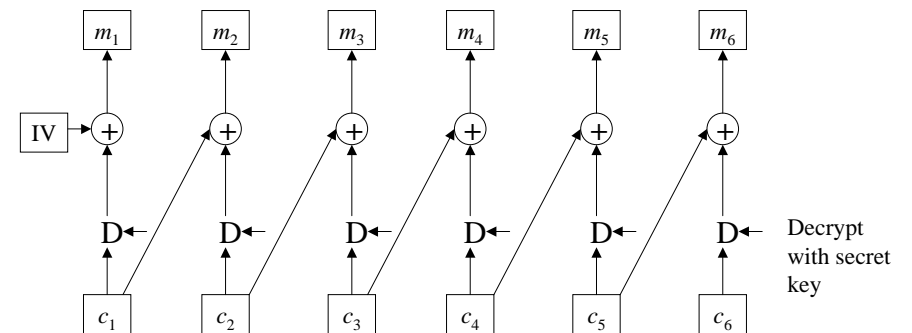
54

## CBC Encryption



55

## CBC Decryption



56

## Modifying Ciphertext Blocks

- of course, the ciphertext blocks can still be modified or rearranged in transit

57

## Output Feedback Mode (OFB)

- acts like a pseudo-random number generator
- message encrypted by bitwise exclusive oring it with the pseudorandom stream generated by OFB

58

## Random Number

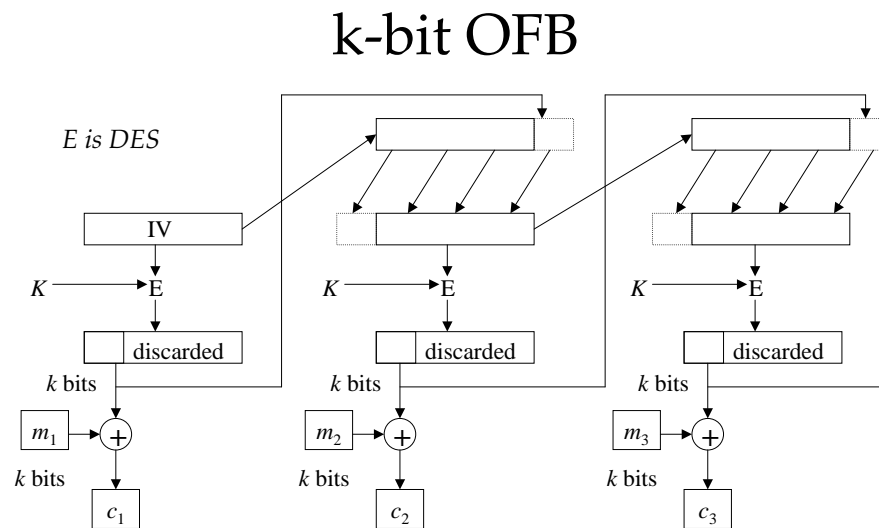
- generate 64 bit random number  $b_0$
- call it the initialisation vector (IV)
- encrypt  $b_0$  with secret key to get  $b_1$
- encrypt  $b_1$  to get  $b_2$  and so on

59

## Encryption & Decryption

- encrypt message by bitwise exclusive oring it with as many bits as necessary of  $b_0 | b_1 | b_2 | b_3 \dots$
- transmit result with IV
- decrypt by again bitwise exclusive oring it with  $b_0 | b_1 | b_2 | b_3 \dots$

60



## One Time Pad

- a long random (or pseudorandom) string
- used to encrypt message with simple bitwise exclusive oring
- known as a **one-time pad**

62

## Advantages

- one time pad can be generated in advance
- makes encryption much quicker
- as actual encryption does not have to be done on-line

63

## Cont.

- if bits of ciphertext garbled, only corresponding bits of plaintext garbled
- of course, this could be a disadvantage
- if message arrives in arbitrary chunks, can be transmitted as arrives
- with CBC must wait until have 64 bits

64

## Disadvantage

- plaintext and ciphertext may be known by attacker
- can modify plaintext by
  - bitwise exclusive oring ciphertext with known plaintext
  - bitwise exclusive oring result with desired result

65

## Cipher Feedback Mode

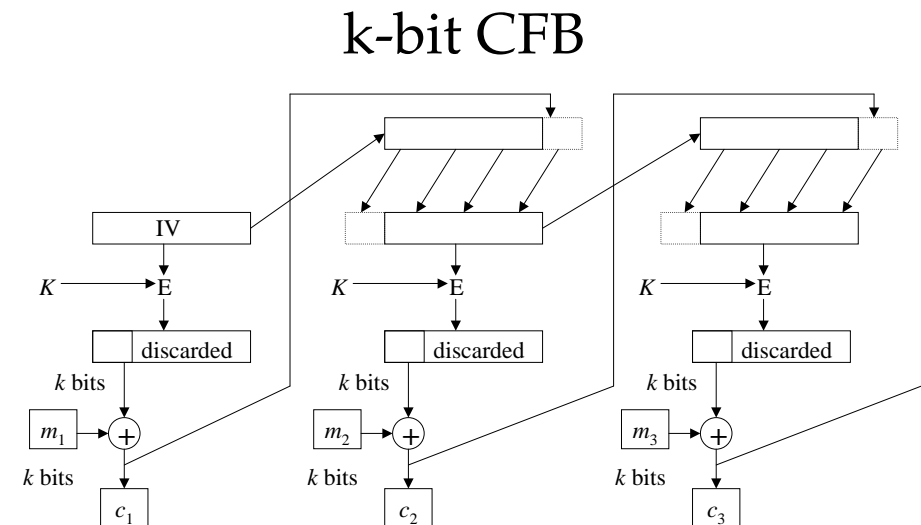
- similar to OFB
- $k$  bits generated and bitwise exclusive or'ed with plaintext
- in OFB  $k$  bits are shifted in to register used as input to DES encrypt are
- output of DES encrypt of previous block

66

## CFB (Cont.)

- in CFB the  $k$  bits shifted in are the  $k$  bits of ciphertext from previous block
- so one time pad cannot be generated before message known

67



68

## Chunk Size

- the  $k$  in  $k$ -bit should not be 64
- 8 bit better
- with OFB or CBC if characters lost or added remainder of transmission garbled
- with 8 bit CFB if error is integral number of bytes things will resynchronise

69

## Tampering

- 8-bit CFB
- attacker can predictably change one byte
- this unpredictably changes next 8 bytes

70

## MICs

- CBC, CFB, OFB offer good protection against eavesdropping
- none offer good protection against an attacker who knows plaintext modifying it

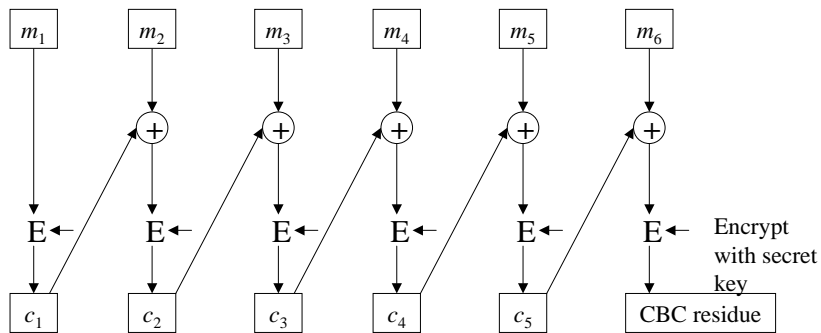
71

## CBC

- compute CBC
- send only last block with plaintext
- last block called **CBC residue**
- to compute residue you must know key
- attacker, not knowing key, cannot modify message and compute corresponding residue

72

## CBC Residue



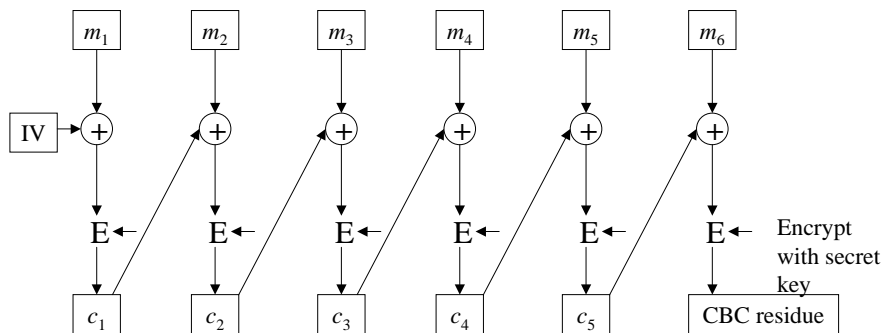
73

## Privacy and Integrity

- CBC encrypt to ensure privacy
- send residue to ensure integrity
- so do both for integrity and privacy?

74

## CBC Encryption and Residue



75

## Hmm

- that just sends encrypted message and repeat final block
- attacker could tamper and send new final block twice
- we want to automatically detect tampering, so need integrity as well as privacy

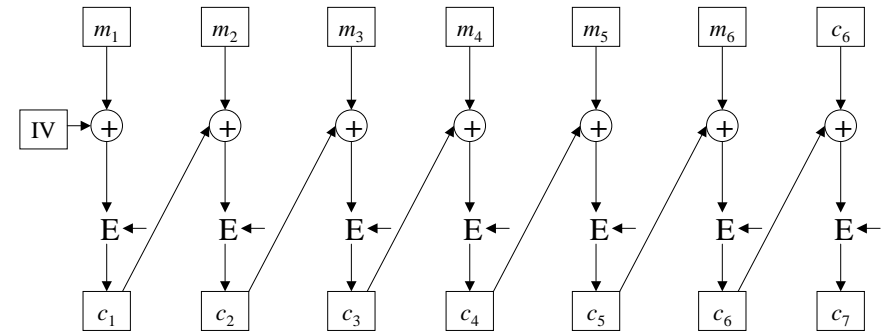
76

## Next Try

- compute CBC residue
- attach that to plaintext
- CBC encrypt result

77

## CBC Encryption Message + Residue



78

## Well

- doesn't work
- last block is encryption of zero
- as anything bitwise exclusive or'ed with itself is zero
- a last block that doesn't depend on the message offers no integrity protection

79

## So?

- can get protection and integrity by using CBC
- encryption and residue calculation using two different keys
- using one key is possible
- but has flaws which may or may not be important depending on situation

80

## Multiple Encryption DES

- due to key length DES not as secure as it could be
- can be made more secure by multiple encryptions
- most accepted method known as EDE (encrypt-decrypt-encrypt)

81

## Multiple Encryption

- can be done for any encryption scheme
- mostly with DES to try to increase its inadequate key length

82

## Encrypt and Decrypt

- functions are inverses of each other
- each actually takes arbitrary data and garbles it in a way that is reversed by other function
- so could perform decrypt on plaintext to encrypt
- then perform encrypt to get plaintext
- let's just call the two functions E and D

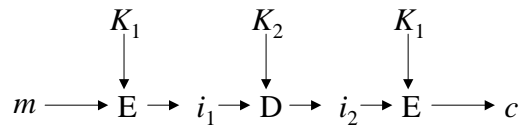
83

## Multiple Encryption

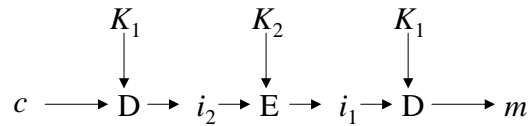
- two keys K1 and K2
- each plaintext block subjected to E with K1
- then D with K2
- then E with K1

84

# Multiple Encryption



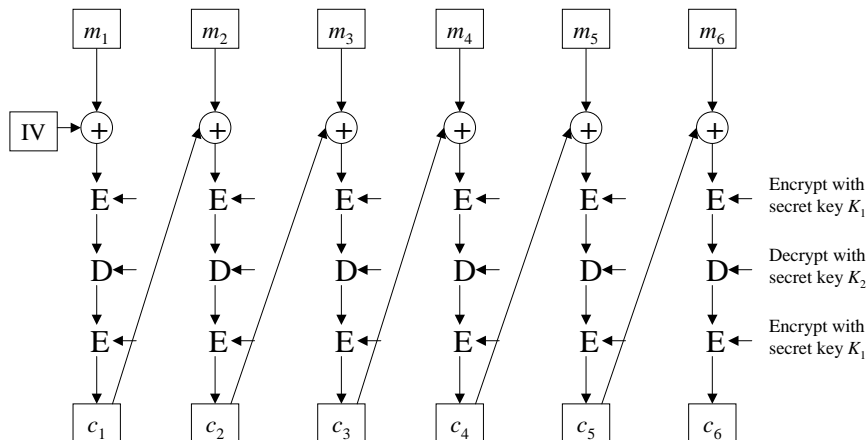
Decryption simply reverses the operation



# Result

- is just a new secret key scheme
- 64 bit input mapped to 64 bit output
- CBC is used to turn block encryption into stream encryption

# EDE



# Discussion

- need more than one key
- what happens if simply encrypt twice using same key?
- exhaustive searching would still only require trying  $2^{56}$  possibilities
- each attempt would require twice as much work, but that's not much

## Discussion

- what about encrypting twice using two different keys?
- not much harder to break than a single key
- assume block encryption
- assume attacker has some plaintext,ciphertext pairs

## Attack

- make two tables of  $2^{56}$  entries for each pair
- first table result of encrypting plaintext
- second table result of decrypting ciphertext
- look for matching entries
- these are possible key pairs
- Takes about twice the time of breaking a 56 bit key