Block Ciphers

Recall: Symmetric-Key Encryption Algorithms

- Also called single-key or secret key algorithms
- Both parties share the key needed to encrypt and decrypt messages, hence both parties are equal
- Modern symmetric key ciphers (developed from product ciphers) include DES, Blowfish, IDEA, LOKI, RC5, Rijndael (AES) and others

Block Ciphers

- One of the most widely used types of cryptographic algorithms
  - For encrypting data to ensure secrecy
  - As a cryptographic checksum to ensure integrity
  - For authentication services
- Used because they are comparatively fast, and we know how to design them
- We'll look at both DES (Data Encryption Standard) and AES (Advanced Encryption Standard)
  - Focus on DES in this slideset

Block vs Stream Ciphers

- Block ciphers process messages in into blocks, each of which is then en/decrypted
  - So all bits of block must be available before processing
- Like a substitution on very big characters
  - 64-bits or more
- Stream ciphers process messages a bit or byte at a time when en/decrypting
  - Though technically the only difference here is block size, there are significant differences in how stream and block ciphers are designed.

Claude Shannon

- Wrote some of the pivotal papers on modern cryptology theory
  - C E Shannon, "Communication Theory of Secrecy Systems", Bell System Technical Journal, Vol 28, Oct 1949, pp 656-715
  - C E Shannon, "Prediction and Entropy of printed English", Bell System Technical Journal, Vol 30, Jan 1951, pp 50-64

Claude Shannon

- Among other things, he developed the concepts of:
  - Entropy of a message
  - Redundancy in a language
  - Theories about how much information is needed to break a cipher
  - Defined the concepts of computationally secure vs unconditionally secure ciphers
  - Introduced the idea of substitution-permutation (S-P) networks, basis of current product ciphers

Shannon S-P Network

- cipher needs to completely obscure statistical properties of original message
  - E.g., a one-time pad does this
- more practically Shannon suggested combining elements to obtain:
  - **diffusion** – dissipates statistical structure of plaintext over bulk of ciphertext
  - **confusion** – makes relationship between ciphertext and key as complex as possible
- S-P networks designed to provide these

Block Cipher Requirements

- Must be reasonably efficient
- Must be able to *efficiently* decrypt ciphertext to recover plaintext
- Must have a reasonable key length
- First attempt: Arbitrary reversible substitution
  - For a large block size this is not practical for implementation and performance reasons

Why Not Arbitrary Reversible Substitution?

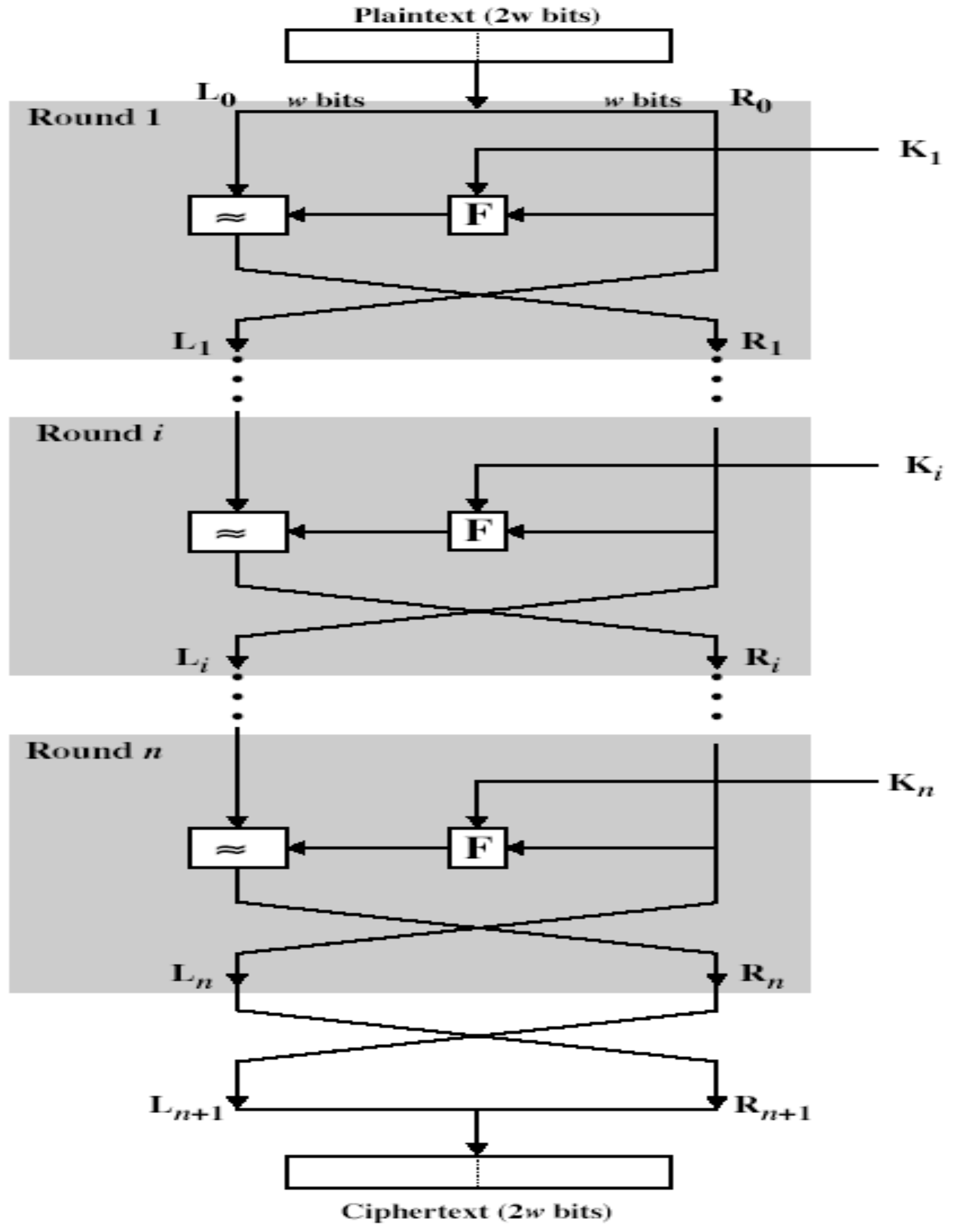
- If we're going from n bit plaintext to n bit ciphertext:
  - There are 2^n possible plaintext blocks.
  - Each must map to a unique output block, so total of $2^n!$ reversible transformations
    - List all n-bit binary (plaintext) strings. First one can go to any of 2^n n-bit binary strings, next to any of 2^n-1 output strings, etc.

Why Not Arbitrary Reversible Substitution?

- If we're going from n bit plaintext to n bit ciphertext:
  - So, to specify a specific transformation, essentially need to provide the list of ciphertext outputs for each input block.
  - How many? Well, 2^n inputs, so 2^n outputs, each n bits long implies an effective key size of $n(2^n)$ bits.
    - For blocks of size 64 (desirable to thwart statistical attacks) this amounts to a key of length $64(2^{64}) = 2^{70} = 2^{67}$ bytes $\sim 1.47 \times 10^{20}$ bytes = 147 TB

Feistel Cipher Structure

- Horst Feistel devised the **Feistel cipher**
  - based on concept of invertible product cipher
  - His main contribution was invention of structure that adapted Shannon’s S-P network into easily inverted structure.
- Process consists of several rounds. In each round:
  - partitions input block into two halves
  - Perform substitution on left half by a *round function* based on right half of data and subkey
  - then have permutation swapping halves
- implements Shannon’s substitution-permutation network concept


Plaintext (2w bits)
L_0
w bits
w bits
R_0
Round 1
K_1
F
L_1
R_1
Round i
K_i
F
L_i
R_i
Round n
K_n
F
L_n
R_n
L_{n+1}
R_{n+1}
Ciphertext (2w bits)

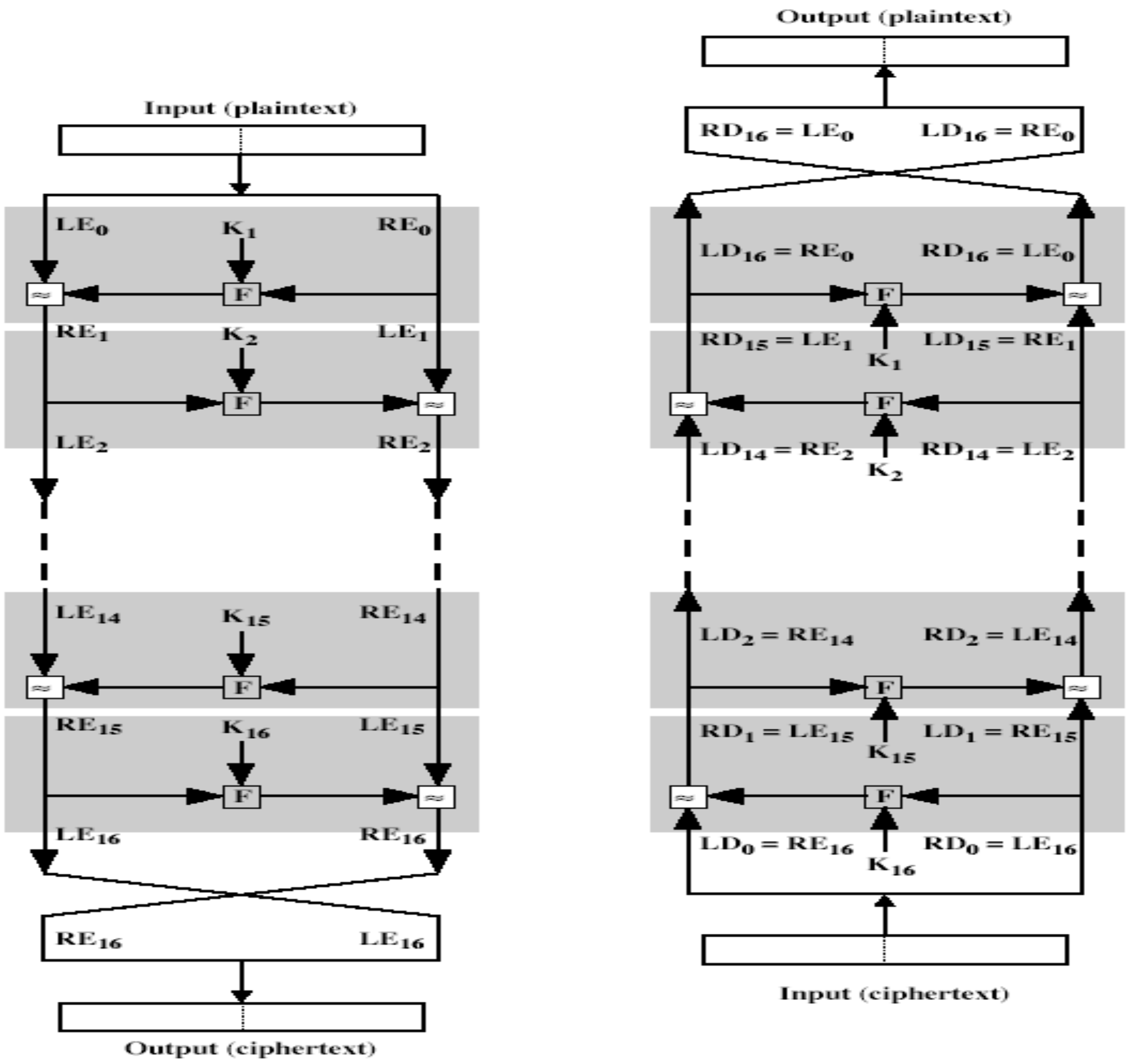
Feistel Cipher Design Principles

- **block size**
  - increasing size improves security, but slows cipher
  - 64 bits reasonable tradeoff. Some use 128 bits
- **key size**
  - increasing size improves security, makes exhaustive key searching harder, but may slow cipher
  - 64 bit considered inadequate. 128 bit is common size (for now)
- **number of rounds**
  - increasing number improves security, but slows cipher

Feistel Cipher Design Principles

- **subkey generation**
  - greater complexity can make analysis harder, but slows cipher
- **round function**
  - greater complexity can make analysis harder, but slows cipher
- **fast software en/decryption & ease of analysis**
  - are more recent concerns for practical use and testing
  - Making algorithms easy to analyze helps determine cipher effectiveness (DES functionality is not easily analyzed)

Feistel Cipher Decryption



Data Encryption Standard (DES)

- was once the most widely used block cipher in world
- adopted in 1977 by NBS (now NIST)
  - as FIPS PUB 46
- encrypts 64-bit data using 56-bit key
- still has widespread use
- At first, considerable controversy over its security
  - Tweaked by NSA?

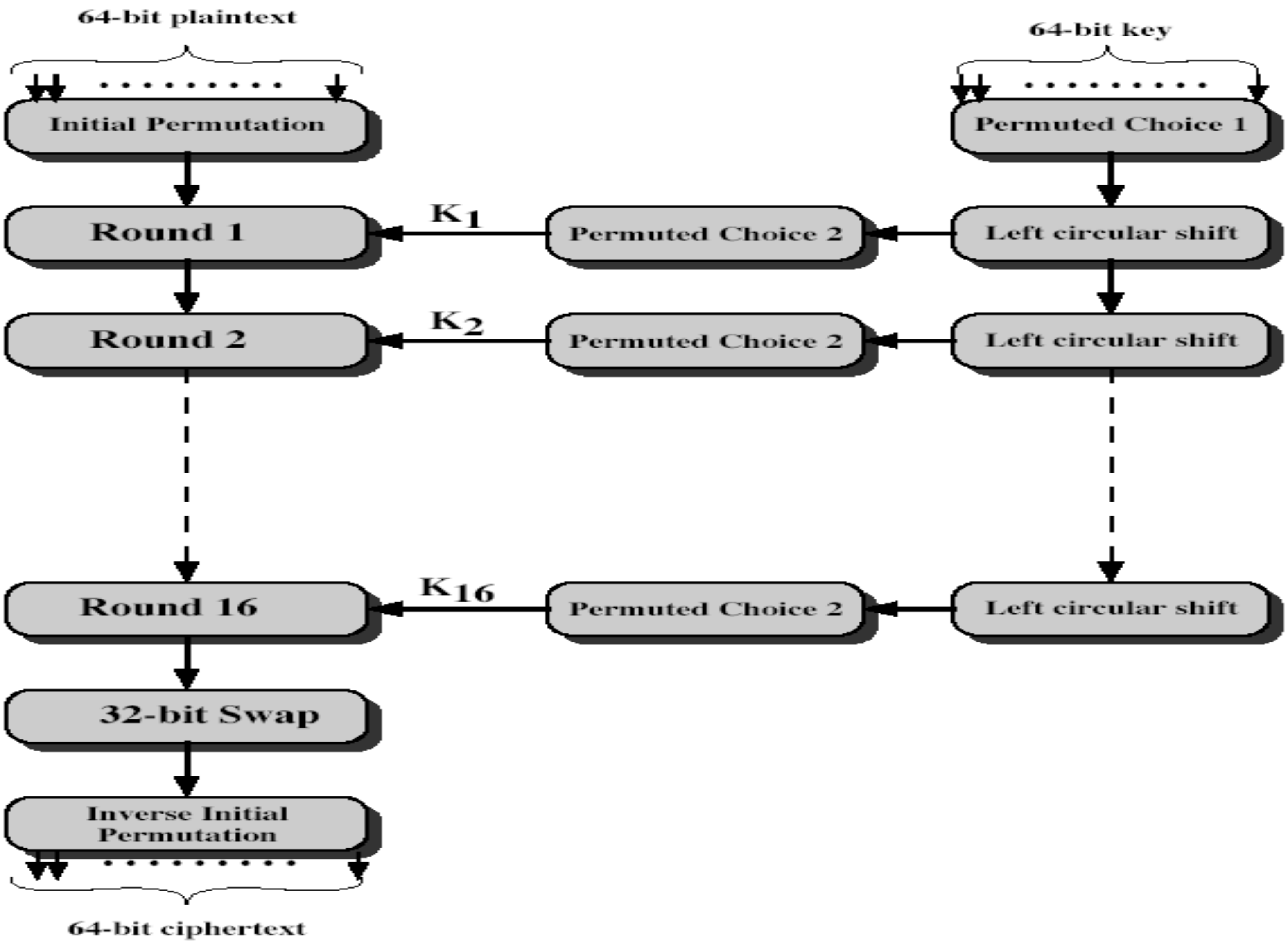
DES History

- IBM developed Lucifer cipher
  - by team led by Feistel
  - used 64-bit data blocks with 128-bit key
- then redeveloped as a commercial cipher with input from NSA and others
- in 1973 NBS issued request for proposals for a national cipher standard
- IBM submitted their revised Lucifer which was eventually accepted as the DES

DES Design Controversy

- Although DES standard is public was considerable controversy over design
  - in choice of 56-bit key (vs Lucifer 128-bit)
  - and because design criteria were classified
  - And because some NSA requested changes incorporated
- Subsequent events and public analysis show in fact design was appropriate
  - Changes made cipher less susceptible to differential or linear cryptanalysis

DES Encryption



Initial Permutation IP

- first step of the data computation
- IP reorders the input data bits
  - Permutation specified by tables (See FIPS 46-3)
- even bits to LH half, odd bits to RH half
- quite regular in structure (easy in h/w)

DES Round Structure

- uses two 32-bit L & R halves
- as for any Feistel cipher can describe as:

  $L_i = R_{i-1}$

  $R_i = L_{i-1}$ xor $F(R_{i-1}, K_i)$
- takes 32-bit R half and 48-bit subkey and:
  - expands R to 48-bits using perm E
  - adds to subkey (XOR)
  - passes through 8 S-boxes to get 32-bit result
    - Each S-box takes 6 bits as input and produces 4 as output
  - finally permutes this using 32-bit perm P

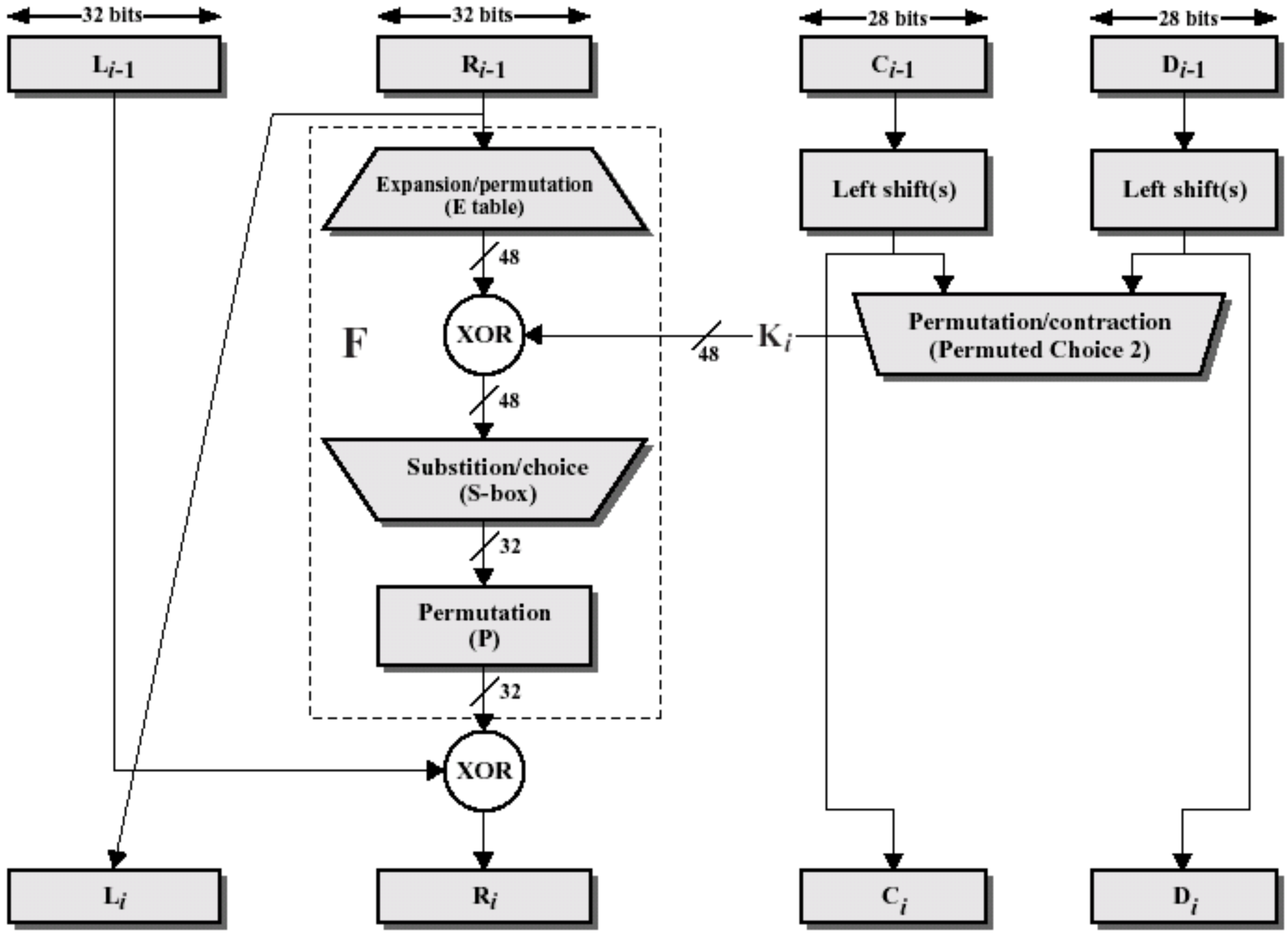
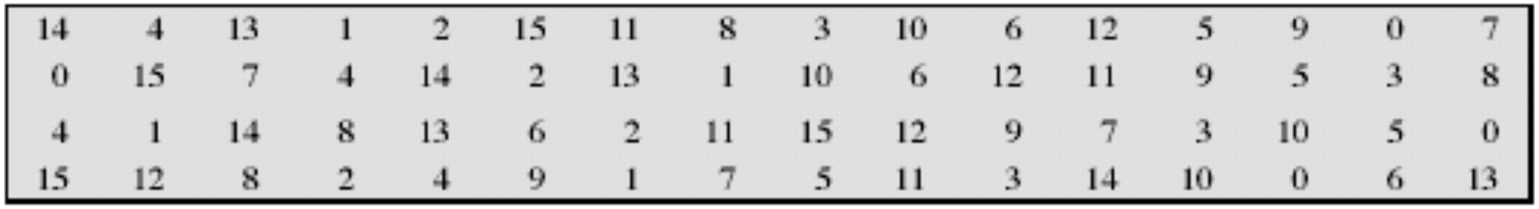


Figure 3.8 Single Round of DES Algorithm

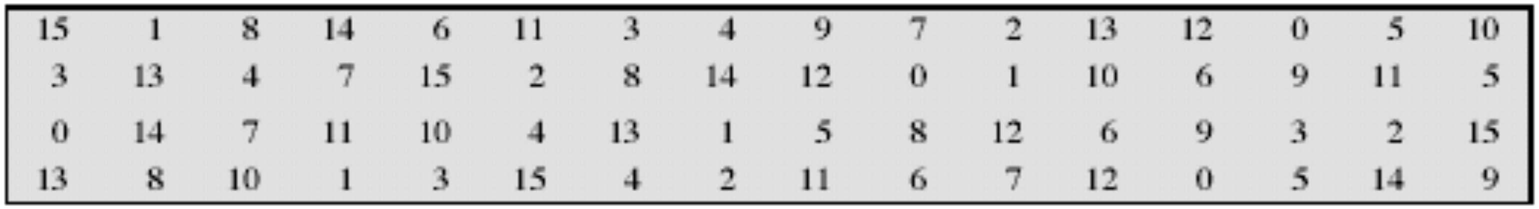
S-boxes

S_1



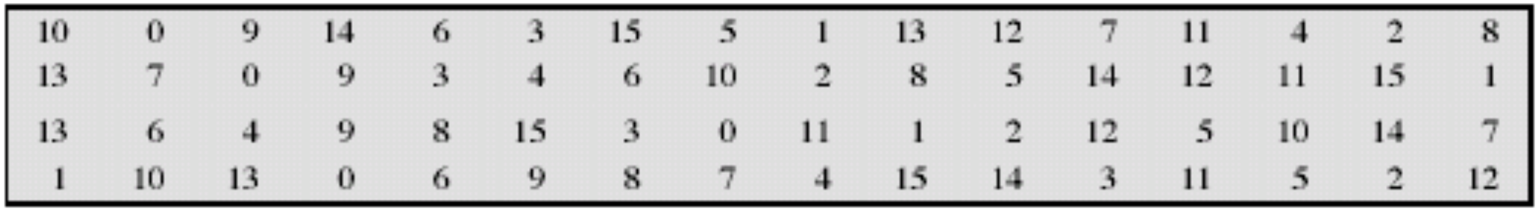
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2



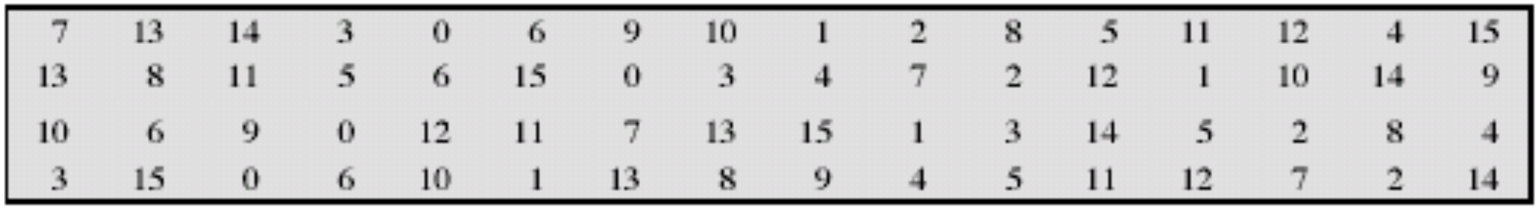
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3



10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

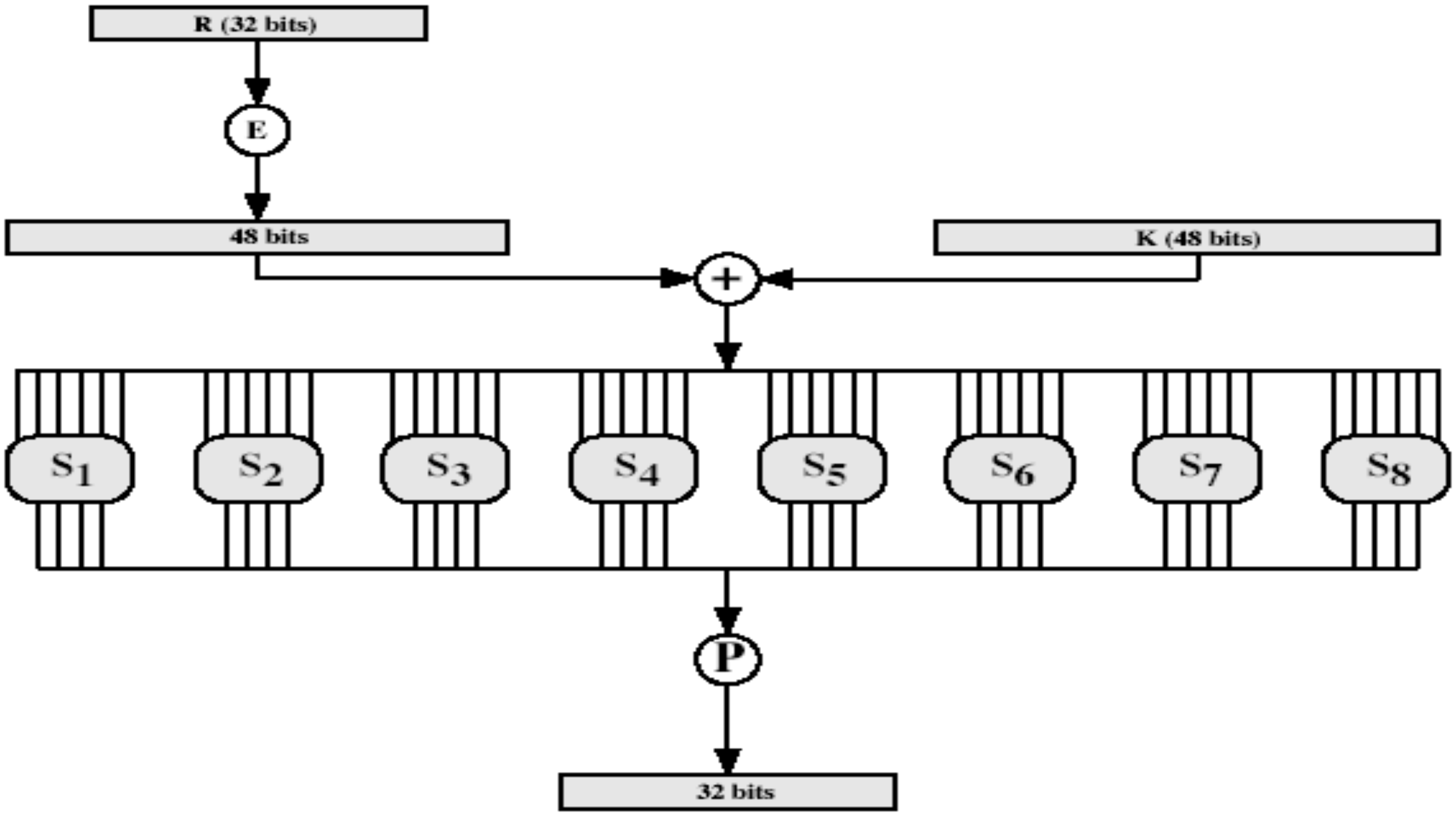
S_4



7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

There are four more

DES Round Structure



Substitution Boxes S

- have eight S-boxes which map 6 to 4 bits
- each S-box is actually 4 little 4 bit boxes
  - outer bits 1 & 6 (**row** bits) considered 2-bit number that selects row
  - inner bits 2-5 (**col** bits) considered 4-bit number that selects column.
  - Decimal number in table is converted to binary and that gives the four output bits
  - result is 8 sets of 4 bits, or 32 bits
- row selection depends on both data & key
  - feature known as autoclaving (autokeying)

DES Key Schedule

- forms subkeys used in each round
- consists of:
  - initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves
  - 16 stages consisting of:
    - selecting 24-bits from each half
    - permuting them by PC2 for use in function f,
    - rotating **each half** separately either 1 or 2 places depending on the **key rotation schedule** K

Table 3.4 DES Key Schedule Calculation

(a) Input Key

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(b) Permuted Choice One (PC-1)

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

(c) Permuted Choice Two (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

(d) Schedule of Left Shifts

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES Decryption

- decrypt must unwind steps of data computation
- with Feistel design, do encryption steps again
- using subkeys in reverse order (SK16 … SK1)
- note that IP undoes final FP step of encryption
- 1st round with SK16 undoes 16th encrypt round
- ….
- 16th round with SK1 undoes 1st encrypt round
- then final FP undoes initial encryption IP
- thus recovering original data value

Avalanche Effect

- Desirable property for an encryption algorithm
- A change of **one** input or key bit results in changing approx **half** output bits
- This makes attempts to "home-in" by guessing keys impossible
- DES exhibits strong avalanche

Strength of DES – Key Size

- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
- brute force search looks hard
- recent advances have shown is possible (as we’ve seen)
  - in 1997: on Internet in a few months
  - in 1998: on dedicated h/w (EFF) in a few days
  - in 1999: above combined in 22hrs!
  - in 2012: 399 seconds on supercomputer
- still must be able to recognize plaintext
- AES has replaced DES as the encryption standard (but DES still widely used)

Strength of DES – Timing Attacks

- attacks actual implementation of cipher
- use knowledge of consequences of implementation to derive knowledge of some/ all subkey bits
- specifically use fact that calculations can take varying times depending on the value of the inputs to it
- particularly problematic on smartcards

Strength of DES – Analytic Attacks

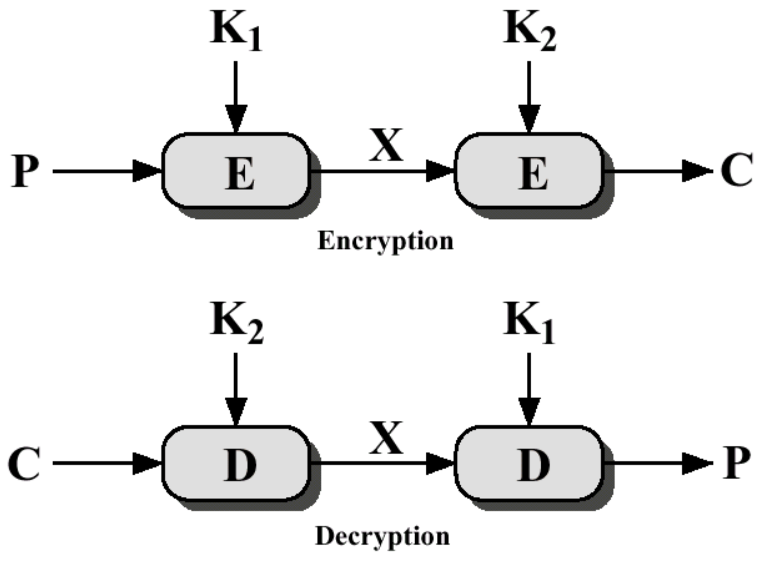
- now have several analytic attacks on DES
- these utilize some deep structure of the cipher
  - by gathering information about encryptions
  - can eventually recover some/all of the sub-key bits
  - if necessary then exhaustively search for the rest
- generally these are statistical attacks
- include
  - differential cryptanalysis
  - linear cryptanalysis

Triple DES

- A replacement for DES was needed
  - theoretical attacks can break it
  - demonstrated exhaustive key search attacks
- AES is a new cipher alternative that didn't exist at the time
- prior to this alternative was to use multiple encryption with DES implementations
- Triple-DES was the chosen form

Why Not Double DES?

- That is, why not just use $C=E_{K1}[E_{K2}[P]]$?
  - Proven that it's NOT same as $C=E_{K3}[P]$
- Susceptible to *Meet-in-the-Middle Attack*
  - Described by Diffie & Hellman in 1977
  - Based on observation that if $C= E_{K2}[E_{K1}[P]]$, then $X=E_{K1}[P]=D_{K2}[C]$



(a) Double Encryption

Meet-in-the-Middle Attack

- Given a known plaintext-ciphertext pair, proceed as follows:
  - Encrypt P for all possible values of K1
    - Cost is on order of 2^{56}
  - Store results in table and sort by value of X
  - Decrypt C for all possible values of K2
    - During each decryption, check table for match. If find one, test two keys against another known plaintext-ciphertext pair

Meet-in-the-Middle Attack

- For any given plaintext P, there are 2^{64} possible ciphertexts produced by Double DES.
- But Double DES effectively has 112 bit key, so there are 2^{112} possible keys.
- On average then, for a given plaintext, the number of different 112 bit keys that will produce a given ciphertext is $2^{112}/2^{64}=2^{48}$
- Thus, first (P,C) pair will produce about 2^{48} false alarms
- Second (P,C) pair, however, reduces false alarm rate to $2^{48-64} = 2^{-16}$. So for two (P,C) pairs, the probability that correct key is determined is $1-(1/2^{16})$.
- Bottom line: a known plaintext attack will succeed against Double DES with an effort on order of 2^{56}, not much more than the 2^{55} required to crack single DES

Triple-DES with Two-Keys

- Would think Triple DES must use 3 encryptions but can use 2 keys with E-D-E sequence
  - $C = E_{K1}[D_{K2}[E_{K1}[P]]]$
  - N.b. encrypt & decrypt equivalent in security
  - if $K1=K2$ then can work with single DES
- standardized in ANSI X9.17 & ISO8732
- no current known practical attacks
  - Though some indications of potential attack strategies, so some use Triple DES with three keys
  - has been adopted by some Internet applications, eg PGP, S/MIME
- Three times slower than DES

Modes of Operation

Modes of Operation

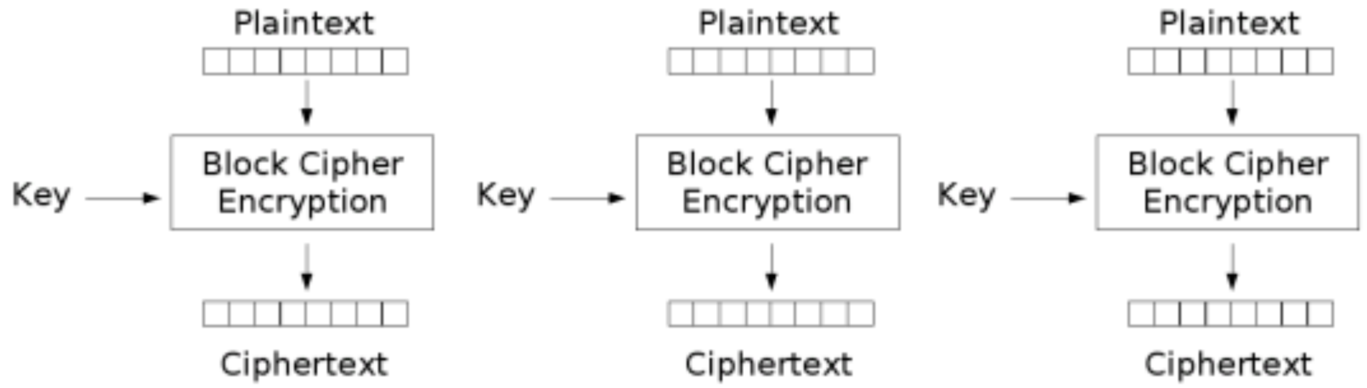
- block ciphers encrypt fixed size blocks
- eg. DES encrypts 64-bit blocks, with 56-bit key
- need way to use in practice, given usually have arbitrary amount of information to encrypt
- four were defined for DES in *DES Modes of Operation*, FIPS PUB 81, in 1981
- subsequently now have 5 for DES and AES
- have **block** and **stream** modes

Electronic Codebook Book (ECB)

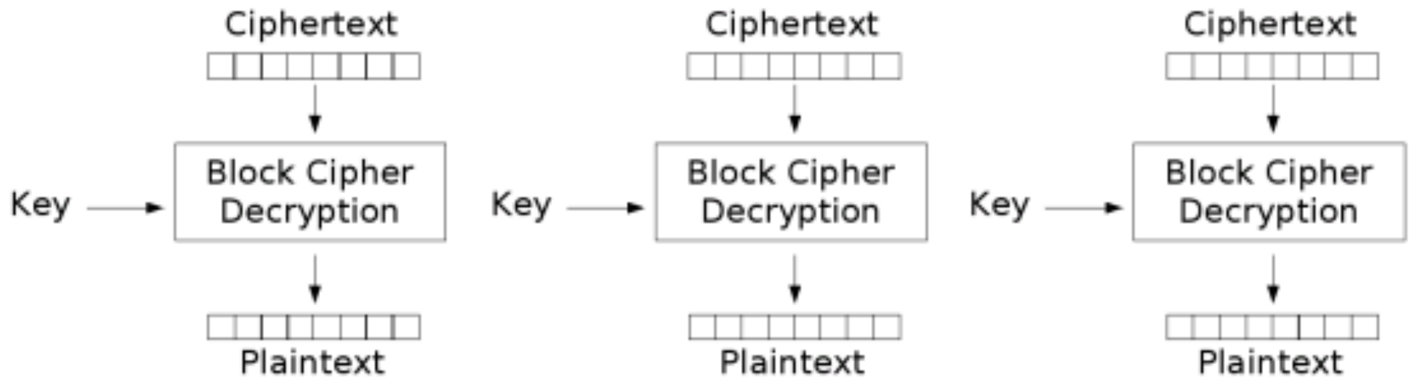
- message is broken into independent blocks which are encrypted
  - Pad last block if necessary to make message length multiple of 64 bits
- each block is a value which is substituted, like a codebook, hence name
- each block is encoded independently of the other blocks

$C_i = DES_{K1} (P_i)$

Electronic Codebook Book (ECB)



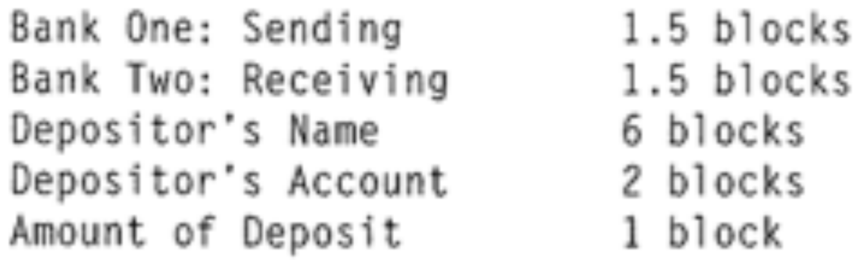
Electronic Codebook (ECB) mode encryption



Electronic Codebook (ECB) mode decryption

Problem (Example from Applied Cryptography. B. Schneier)

- Adversary can modify encrypted messages without knowing the key or even the algorithm in manner that fools recipient
- Example: Money transfer between banks
  - Assume an agreed standard message format (below)



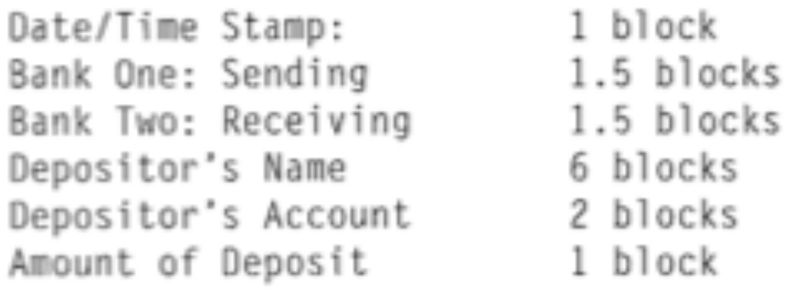
Bank One: Sending	1.5 blocks
Bank Two: Receiving	1.5 blocks
Depositor's Name	6 blocks
Depositor's Account	2 blocks
Amount of Deposit	1 block

Problem (cont.)

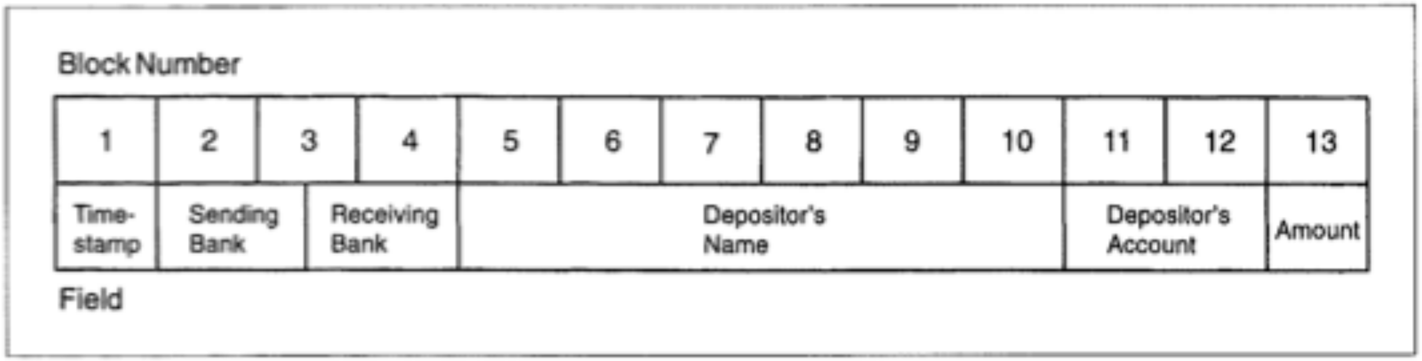
- Transfers encrypted using some block cipher in ECB mode
- Trudy, listens on communication lines between Bank of Alice and Bank of Bob
- She opens accounts at both banks, and transfers $100 from her account at Bank of Alice to her account at Bank of Bob. Twice.
- Checks communication records to find two identical messages (presumably her transfer)
- Inserts copies of her transfer into communication link at will!
  - If clever, done with large amounts and many banks!

Solution?: Timestamp

- Bank adds timestamp to messages so they can't be replayed:

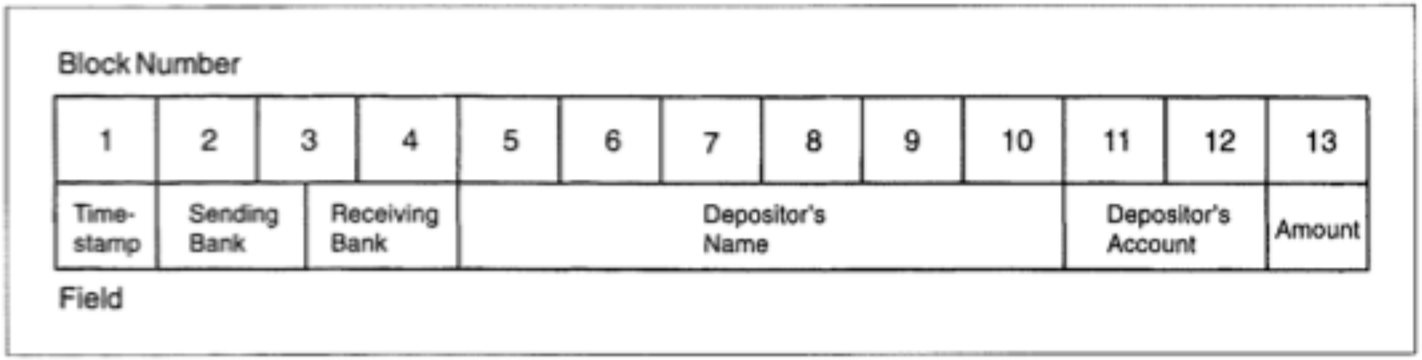


Date/Time Stamp:	1 block
Bank One: Sending	1.5 blocks
Bank Two: Receiving	1.5 blocks
Depositor's Name	6 blocks
Depositor's Account	2 blocks
Amount of Deposit	1 block



Solution?: Timestamp

- Trudy then sends multiple messages, this time examining blocks 5 through 12
- She replaces blocks 5 - 12 of many transfers with her ciphertext blocks 5 - 12!
  - This one won't be caught nearly as quickly (since banks books will still balance at end of day)!



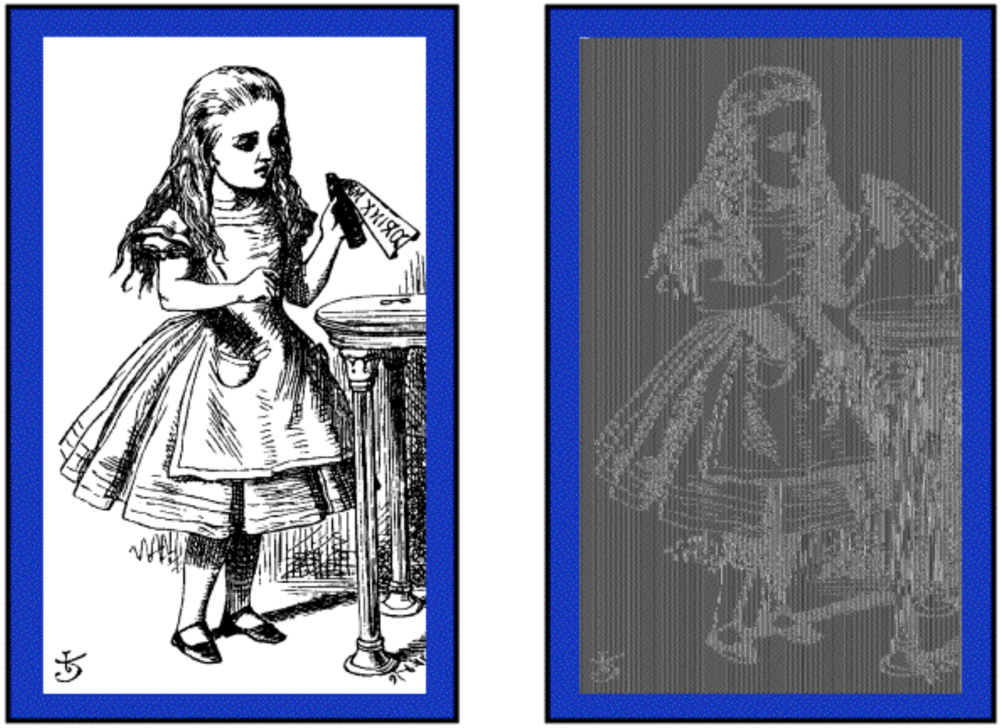
Block Number

1	2	3	4	5	6	7	8	9	10	11	12	13
Time-stamp	Sending Bank		Receiving Bank	Depositor's Name						Depositor's Account		Amount

Field

Graphics Issue with ECB

- Why does this happen (thanks to unknown colleague at San Jose State)?



Bottom Line:

- repetitions in message may show in ciphertext
  - if aligned with message block
  - particularly with data such as graphics
  - or with messages that change very little, which become a code-book analysis problem
- weakness due to encrypted message blocks being independent
- Attacker can reorder cipher blocks in transit
  - or perhaps even insert or replace a block
- main use is sending a few blocks of data
  - E.g. Transmitting an encryption key

ECB: Advantages

- Encryption is not serial, so it can be done on individual blocks regardless of location in file
  - E.g., large data file
- Encryption/Decryption can be parallelized
- bit error in one ciphertext block does not prevent decryption of other blocks

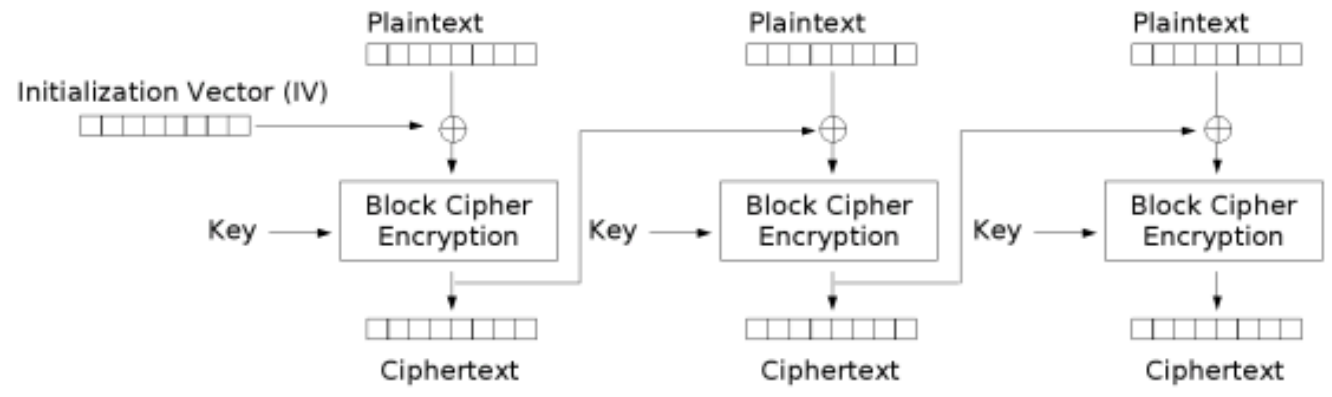
Cipher Block Chaining (CBC)

- Wanted a method in which repeated blocks of plaintext (and whole messages) are encrypted differently each time
- Like ECB, message is broken into blocks, but these are linked together in the encryption operation
- each previous cipher blocks is chained with current plaintext block, hence name
- use Initial Vector (IV) to start process

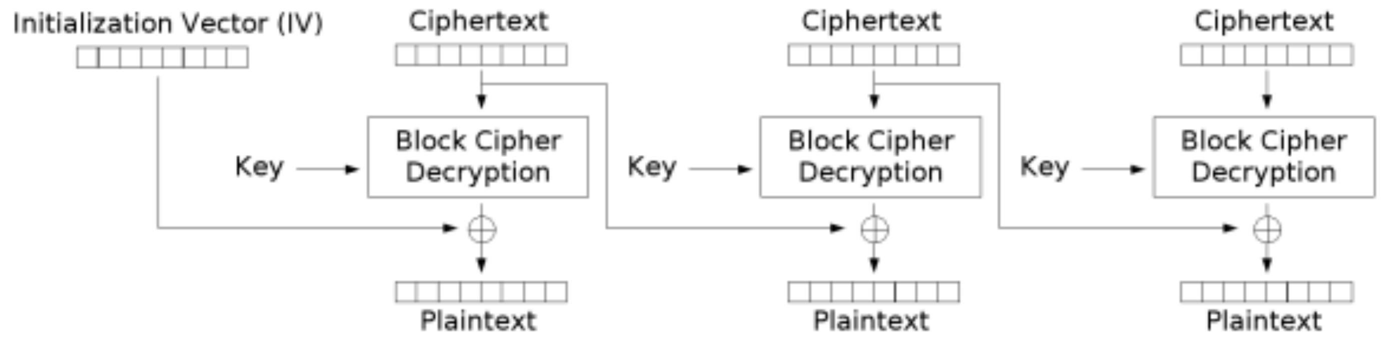
  $C_i = DES_{K1}(P_i \text{ XOR } C_{i-1})$

  $C_{-1} = IV$
- Used for bulk data encryption, authentication

Cipher Block Chaining (CBC)



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

CBC Decryption

$$C_j = E_K[C_{j-1} \oplus P_j]$$

Encryption step

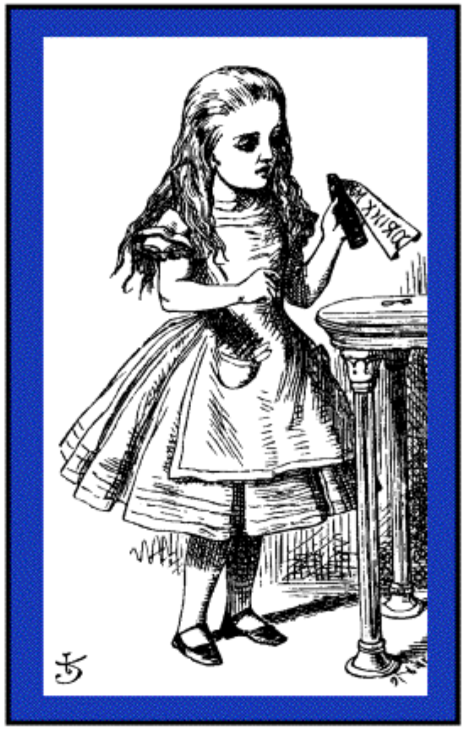
Decryption step (with justification)

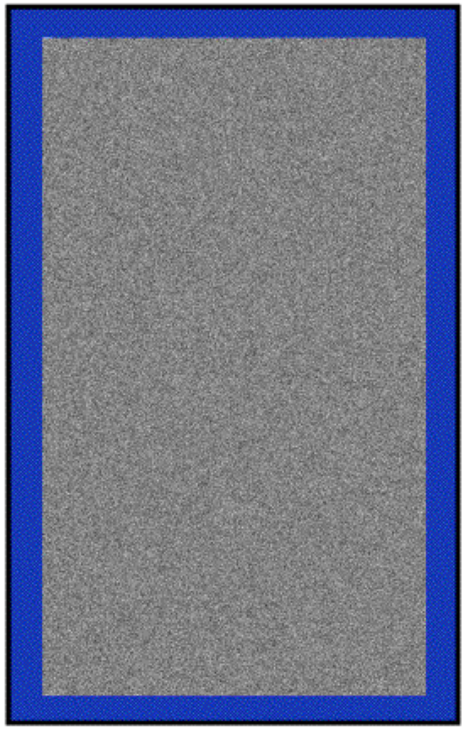
$$D_K[C_j] = D_K[E_K(C_{j-1} \oplus P_j)]$$

$$D_K[C_j] = (C_{j-1} \oplus P_j)$$

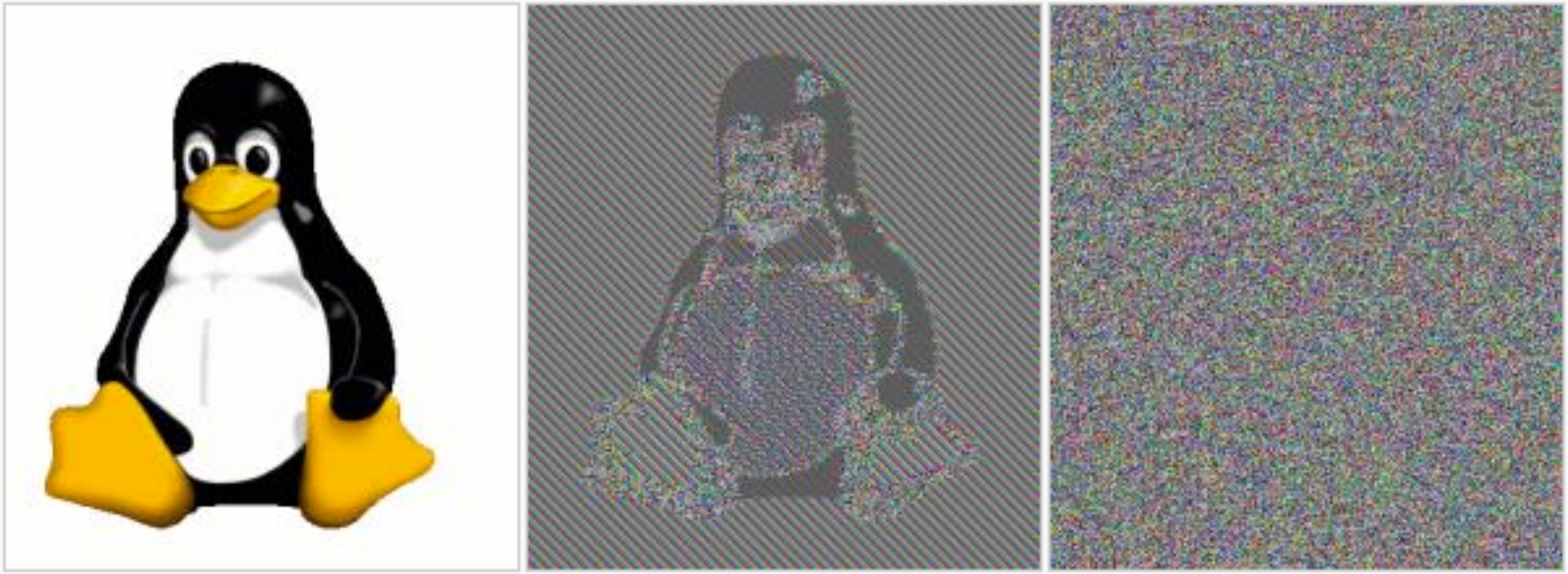
$$C_{j-1} \oplus D_K[C_j] = C_{j-1} \oplus C_{j-1} \oplus P_j = P_j$$

Graphics with CBC





Another Graphics Example (thanks Wikipedia)



Original image

Encrypted using ECB mode

Modes other than ECB result in pseudo-randomness

The image on the right is how the image might appear encrypted with CBC, CTR or any of the other more secure modes—indistinguishable from random noise. Note that the random appearance of the image on the right does not ensure that the image has been securely encrypted; many kinds of insecure encryption have been developed which would produce output just as 'random-looking'.

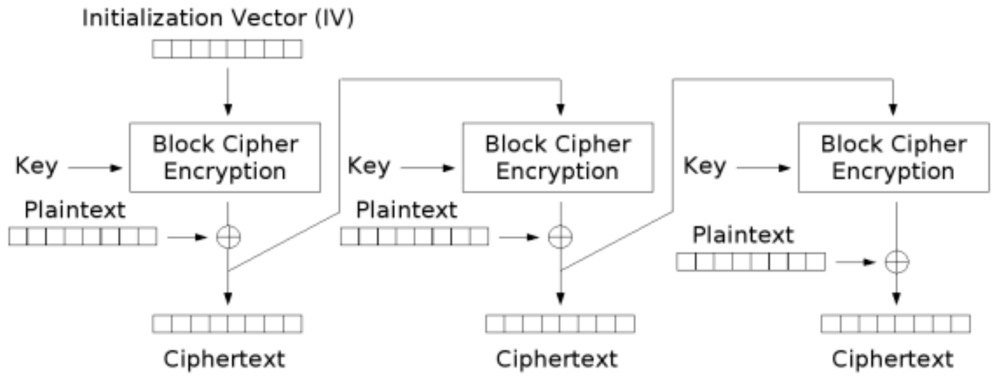
Advantages and Limitations of CBC

- Good: each ciphertext block depends on **all** message blocks, thus a change in the message affects all ciphertext blocks after the change as well as the original block
- need **Initial Value** (IV) known to sender & receiver
  - however if IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate
  - hence either IV must be a fixed value or it must be sent encrypted in ECB mode before rest of message
  - Note that randomly chosen IV means attacker cannot supply known plaintext to underlying cipher even if they can supply plaintext to CBC
  - Weak IV was cause of weakness of WEP

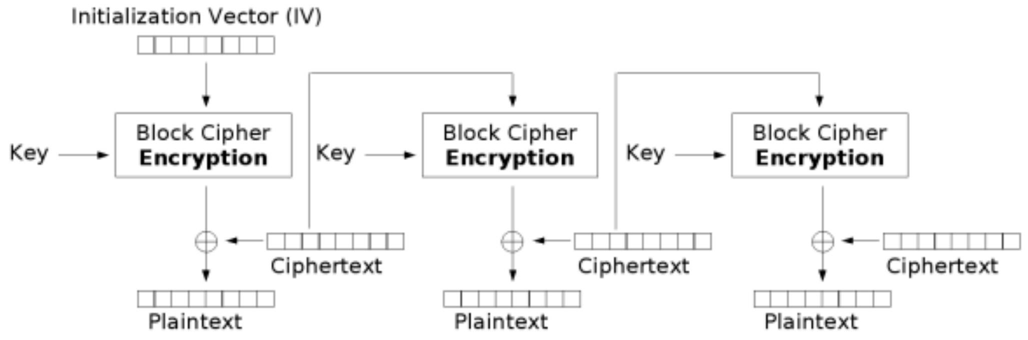
Integrity!

- An important issue related to the choice of IV is that of integrity
- Remember: Confidentiality is not integrity!
  - If you want to guarantee the message has not been tampered with, use an integrity mechanism along with an encryption scheme.

There are Other Modes...



Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

- See Wiki article on block cipher modes of operation