Department of Electronics & Communication Engg.

Course : Network Security **Sem.: 8th (2017-18 EVEN)**

Course Coordinator:

Prof. Nyamatulla M Patel

CRYPTOGRAPHY AND NETWORK SECURITY

Unit-02

Classical Encryption Techniques

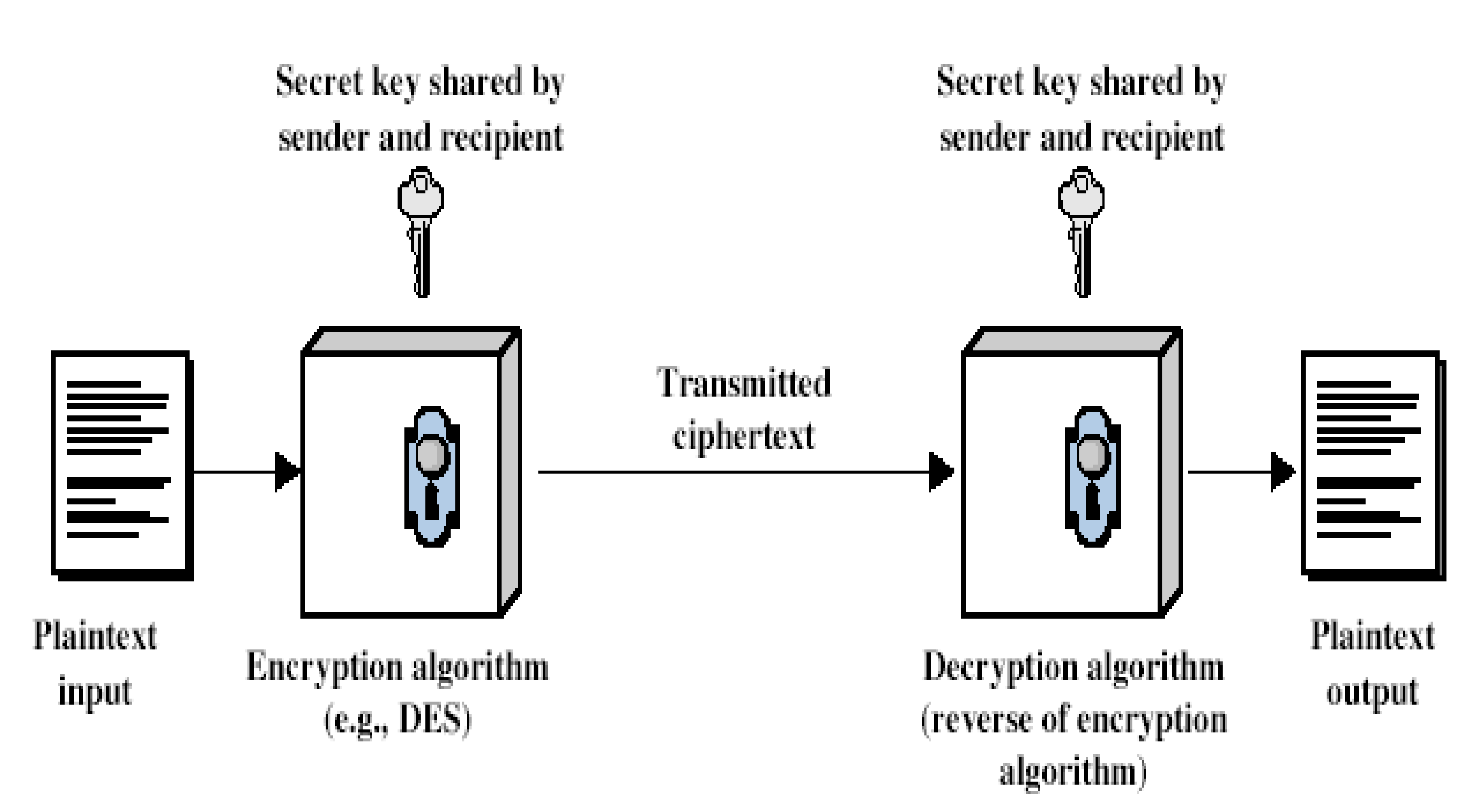
Symmetric Encryption

- or conventional / private-key / single-key
- sender and recipient share a common key
- all classical encryption algorithms are private-key
- was only type prior to invention of public-key in 1970's
- and by far most widely used

Some Basic Terminology

- **plaintext** - original message
- **ciphertext** - coded message
- **cipher** - algorithm for transforming plaintext to ciphertext
- **key** - info used in cipher known only to sender/receiver
- **encipher (encrypt)** - converting plaintext to ciphertext
- **decipher (decrypt)** - recovering ciphertext from plaintext
- **cryptography** - study of encryption principles/methods
- **cryptanalysis (codebreaking)** - study of principles/ methods of deciphering ciphertext *without* knowing key
- **cryptology** - field of both cryptography and cryptanalysis

Symmetric Cipher Model



Requirements

- two requirements for secure use of symmetric encryption:
  - a strong encryption algorithm
  - a secret key known only to sender / receiver
- mathematically have:

$$Y = \mathrm{E}_K(X)$$

$$X = \mathrm{D}_K(Y)$$

- assume encryption algorithm is known
- implies a secure channel to distribute key

Cryptography

- characterize cryptographic system by:
  - type of encryption operations used
    - substitution / transposition / product
  - number of keys used
    - single-key or private / two-key or public
  - way in which plaintext is processed
    - block / stream

Cryptanalysis

- objective to recover key not just message
- general approaches:
  - cryptanalytic attack
  - brute-force attack

Cryptanalytic Attacks

- **ciphertext only**
  - only knows algorithm & ciphertext
- **known plaintext**
  - know/suspect plaintext & ciphertext
- **chosen plaintext**
  - select plaintext and obtain ciphertext
- **chosen ciphertext**
  - select ciphertext and obtain plaintext
- **chosen text**
  - select plaintext or ciphertext to en/decrypt

More Definitions

- **unconditional security**
  - no matter how much computer power or time is available, the cipher cannot be broken since the ciphertext provides insufficient information to uniquely determine the corresponding plaintext
- **computational security**
  - given limited computing resources (eg time needed for calculations is greater than age of universe), the cipher cannot be broken

Brute Force Search

- always possible to simply try every key
- most basic attack, proportional to key size
- assume either know / recognise plaintext

Key Size (bits)	Number of Alternative Keys	Time required at 1 decryption/µs	Time required at 10^{6} decryptions/µs
32	$2^{32} = 4.3 \square 10^{9}$	2^{31} µs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \square 10^{16}$	2^{55} µs = 1142 years	10.01 hours
128	$2^{128} = 3.4 \square 10^{38}$	2^{127} µs = $5.4 \square 10^{24}$ years	$5.4 \square 10^{18}$ years
168	$2^{168} = 3.7 \square 10^{50}$	2^{167} µs = $5.9 \square 10^{36}$ years	$5.9 \square 10^{30}$ years
26 characters (permutation)	$26! = 4 \square 10^{26}$	$2 \square 10^{26}$ µs = $6.4 \square 10^{12}$ years	$6.4 \square 10^{6}$ years

Classical Substitution Ciphers

- where letters of plaintext are replaced by other letters or by numbers or symbols
- or if plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns

Caesar Cipher

- earliest known substitution cipher
- by Julius Caesar
- first attested use in military affairs
- replaces each letter by 3rd letter on
- example:

meet me after the toga party

PHHW PH DIWHU WKH WRJD SDUWB

Caesar Cipher

- can define transformation as:

a b c d e f g h i j k l m n o p q r s t u v w x y z

D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

- mathematically give each letter a number

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	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

- then have Caesar cipher as:

$c = \mathrm{E}(p) = (p + k) \bmod (26)$

$p = \mathrm{D}(\mathrm{c}) = (\mathrm{c} - k) \bmod (26)$

Cryptanalysis of Caesar Cipher

- only have 26 possible ciphers
  - A maps to A,B,..Z
- could simply try each in turn
- a **brute force search**
- given ciphertext, just try all shifts of letters
- do need to recognize when have plaintext
- eg. break ciphertext "GCUA VQ DTGCM"

Monoalphabetic Cipher

- rather than just shifting the alphabet
- could shuffle (jumble) the letters arbitrarily
- each plaintext letter maps to a different random ciphertext letter
- hence key is 26 letters long

Plain: abcdefghijklmnopqrstuvwxyz
Cipher: DKVQFIBJWPESCXHTMYAUOLRGZN

Plaintext: ifwewishtoreplaceletters
Ciphertext: WIRFRWAJUHYFTSDVFSFUUFYA

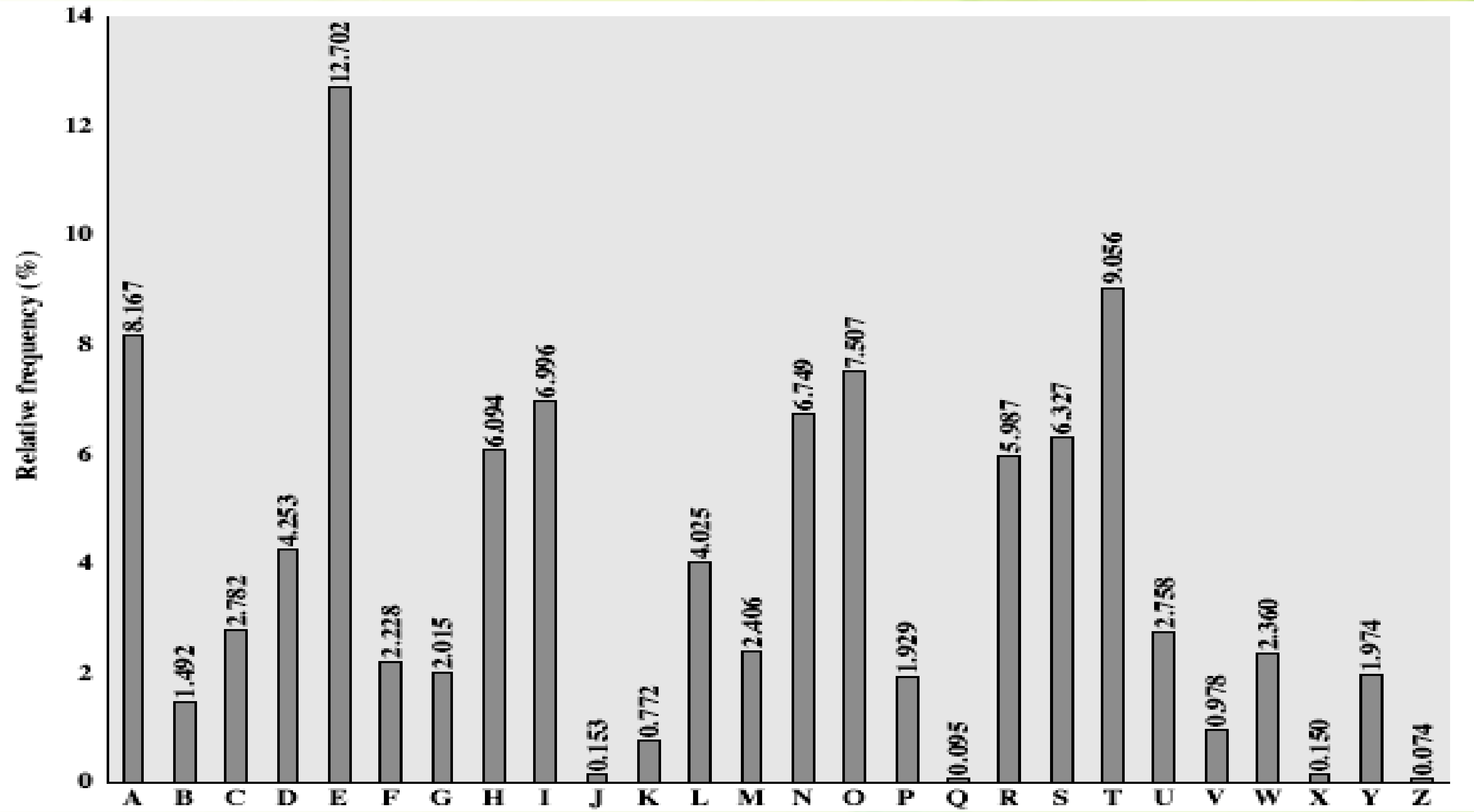
Monoalphabetic Cipher Security

- now have a total of 26! = 4 x 1026 keys
- with so many keys, might think is secure
- but would be **!!!WRONG!!!**
- problem is language characteristics

Language Redundancy and Cryptanalysis

- human languages are **redundant**
- eg "th lrd s m shphrd shll nt wnt"
- letters are not equally commonly used
- in English E is by far the most common letter
  - followed by T,R,N,I,O,A,S
- other letters like Z,J,K,Q,X are fairly rare
- have tables of single, double & triple letter frequencies for various languages

English Letter Frequencies



Use in Cryptanalysis

- key concept - monoalphabetic substitution ciphers do not change relative letter frequencies
- discovered by Arabian scientists in 9th century
- calculate letter frequencies for ciphertext
- compare counts/plots against known values
- if caesar cipher look for common peaks/troughs
  - peaks at: A-E-I triple, NO pair, RST triple
  - troughs at: JK, X-Z
- for monoalphabetic must identify each letter
  - tables of common double/triple letters help

Polyalphabetic Ciphers

- **polyalphabetic substitution ciphers**
- improve security using multiple cipher alphabets
- make cryptanalysis harder with more alphabets to guess and flatter frequency distribution
- use a key to select which alphabet is used for each letter of the message
- use each alphabet in turn
- repeat from start after end of key is reached

Vigenère Cipher

- simplest polyalphabetic substitution cipher
- effectively multiple caesar ciphers
- key is multiple letters long $K = k_1\ k_2\ \ldots\ k_d$
- i^{th} letter specifies i^{th} alphabet to use
- use each alphabet in turn
- repeat from start after d letters in message
- decryption simply works in reverse

Example of Vigenère Cipher

- write the plaintext out
- write the keyword repeated above it
- use each key letter as a caesar cipher key
- encrypt the corresponding plaintext letter
- eg using keyword *deceptive*

key: deceptivedeceptivedeceptive

plaintext: wearediscoveredsaveyourself

ciphertext:ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Security of Vigenère Ciphers

- have multiple ciphertext letters for each plaintext letter
- hence letter frequencies are obscured
- but not totally lost
- start with letter frequencies
  - see if look monoalphabetic or not
- if not, then need to determine number of alphabets, since then can attach each

One-Time Pad

- if a truly random key as long as the message is used, the cipher will be secure
- called a One-Time pad
- is unbreakable since ciphertext bears no statistical relationship to the plaintext
- since for **any plaintext** & **any ciphertext** there exists a key mapping one to other
- can only use the key **once** though
- problems in generation & safe distribution of key

Transposition Ciphers

- now consider classical **transposition** or **permutation** ciphers
- these hide the message by rearranging the letter order
- without altering the actual letters used
- can recognise these since have the same frequency distribution as the original text

Rail Fence cipher

- write message letters out diagonally over a number of rows
- then read off cipher row by row
- eg. write message out as:

m e m a t r h t g p r y
 e t e f e t e o a a t

- giving ciphertext

MEMATRHTGPRYETEFETEOAAT

Row Transposition Ciphers

- a more complex transposition
- write letters of message out in rows over a specified number of columns
- then reorder the columns according to some key before reading off the rows

Key: 3 4 2 1 5 6 7
Plaintext: a t t a c k p
o s t p o n e
d u n t i l t
w o a m x y z
Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

Product Ciphers

- ciphers using substitutions or transpositions are not secure because of language characteristics
- hence consider using several ciphers in succession to make harder, but:
  - two substitutions make a more complex substitution
  - two transpositions make more complex transposition
  - but a substitution followed by a transposition makes a new much harder cipher
- this is bridge from classical to modern ciphers

Rotor Machines

- before modern ciphers, rotor machines were most common complex ciphers in use
- widely used in WW2
  - German Enigma, Allied Hagelin, Japanese Purple
- implemented a very complex, varying substitution cipher
- used a series of cylinders, each giving one substitution, which rotated and changed after each letter was encrypted
- with 3 cylinders have 26^3=17576 alphabets

Hagelin Rotor Machine

Summary

- have considered:
  - classical cipher techniques and terminology
  - monoalphabetic substitution ciphers
  - cryptanalysis using letter frequencies
  - polyalphabetic ciphers
  - transposition ciphers
  - product ciphers and rotor machines

CRYPTOGRAPHY AND NETWORK SECURITY

Unit-02 (Continued…)

Block Ciphers and the Data Encryption Standard

Block Ciphers and the Data Encryption Standard

All the afternoon Mungo had been working on Stern's code, principally with the aid of the latest messages which he had copied down at the Nevin Square drop. Stern was very confident. He must be well aware London Central knew about that drop. It was obvious that they didn't care how often Mungo read their messages, so confident were they in the impenetrability of the code.

—*Talking to Strange Men,* Ruth Rendell

Modern Block Ciphers

- will now look at modern block ciphers
- one of the most widely used types of cryptographic algorithms
- provide secrecy and/or authentication services
- in particular will introduce DES (Data Encryption Standard)

Block vs Stream Ciphers

- block ciphers process messages in into blocks, each of which is then en/decrypted
- 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
- many current ciphers are block ciphers
- hence are focus of course

Block Cipher Principles

- most symmetric block ciphers are based on a **Feistel Cipher Structure**
- needed since must be able to **decrypt** ciphertext to recover messages efficiently
- block ciphers look like an extremely large substitution
- would need table of 2^{64} entries for a 64-bit block
- instead create from smaller building blocks
- using idea of a product cipher

Claude Shannon and Substitution-Permutation Ciphers

- in 1949 Claude Shannon introduced idea of substitution-permutation (S-P) networks
  - modern substitution-transposition product cipher
- these form the basis of modern block ciphers
- S-P networks are based on the two primitive cryptographic operations we have seen before:
  - *substitution* (S-box)
  - *permutation* (P-box)
- provide *confusion* and *diffusion* of message

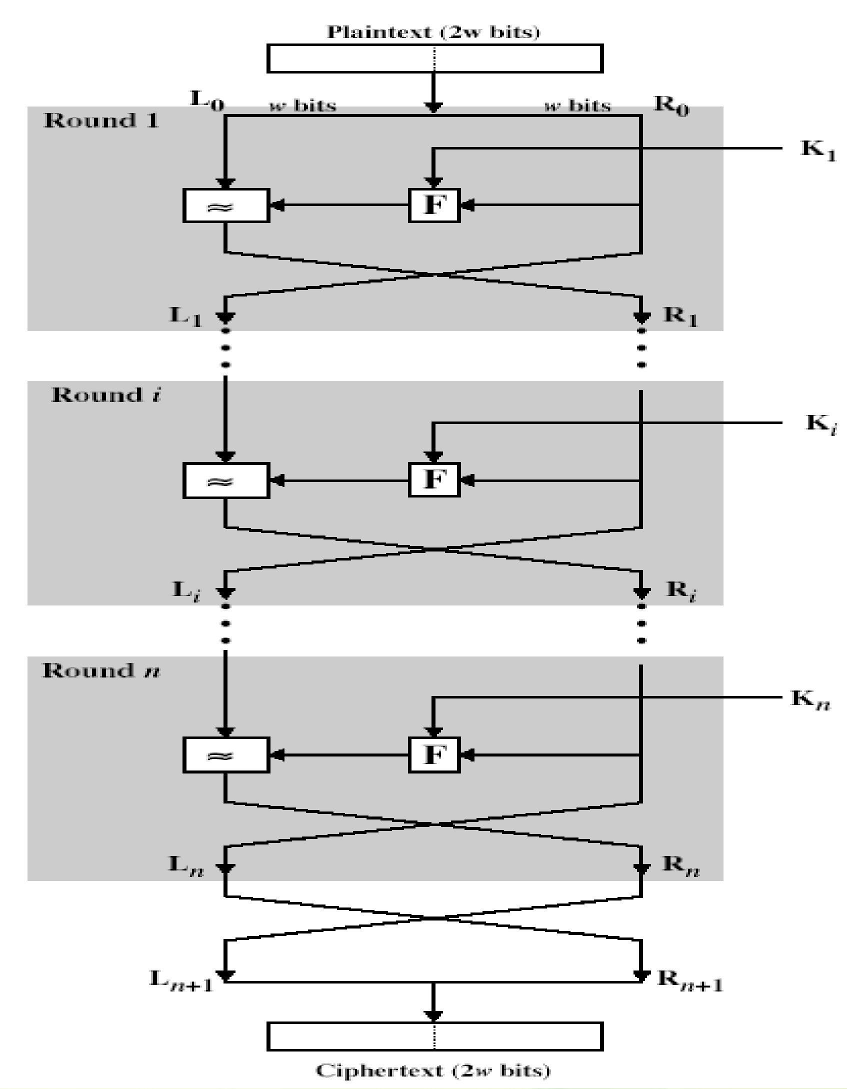
Confusion and Diffusion

- cipher needs to completely obscure statistical properties of original message
- 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

Feistel Cipher Structure

- Horst Feistel devised the **feistel cipher**
  - based on concept of invertible product cipher
- partitions input block into two halves
  - process through multiple rounds which
  - perform a substitution on left data half
  - based on round function of right half & subkey
  - then have permutation swapping halves
- implements Shannon's substitution-permutation network concept

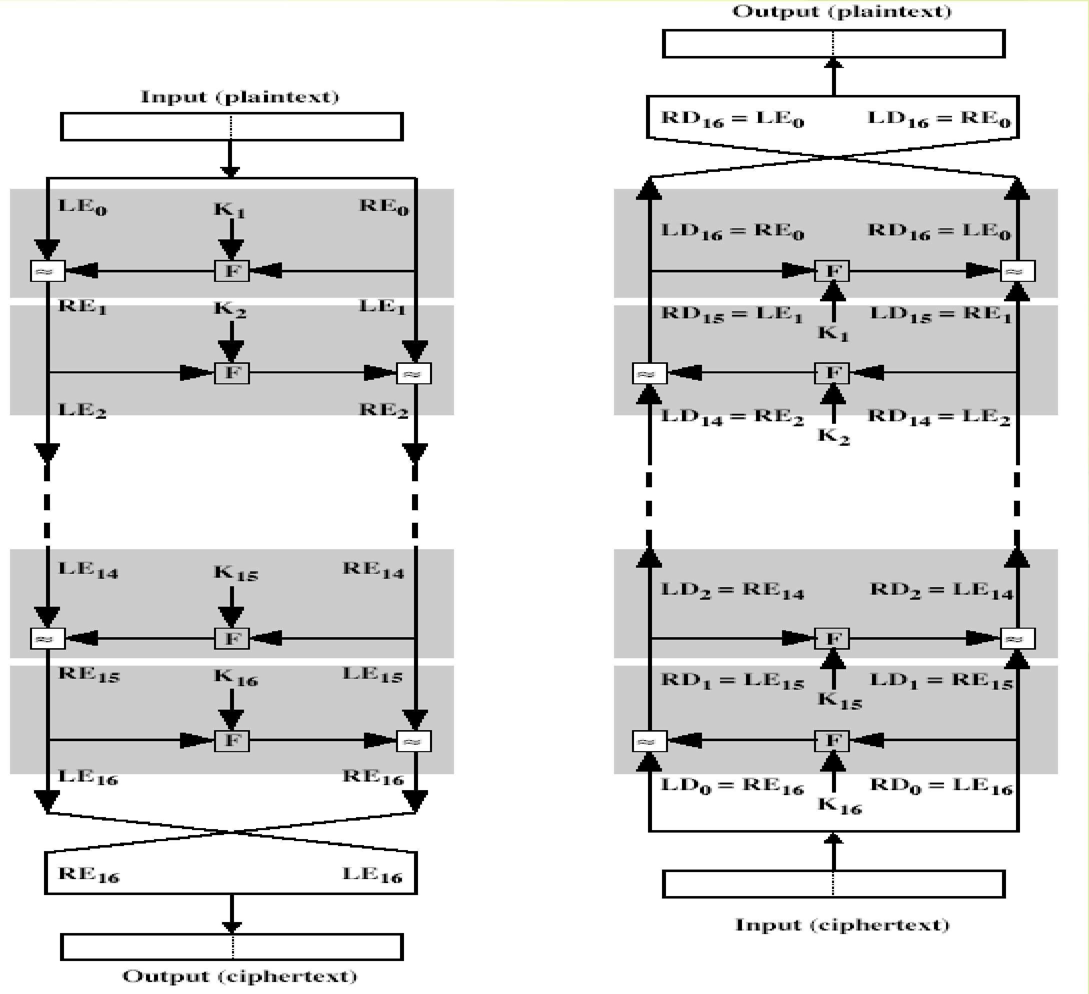
Feistel Cipher Structure



Feistel Cipher Design Principles

- **block size**
  - increasing size improves security, but slows cipher
- **key size**
  - increasing size improves security, makes exhaustive key searching harder, but may slow cipher
- **number of rounds**
  - increasing number improves security, but slows cipher
- **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

Feistel Cipher Decryption



Data Encryption Standard (DES)

- 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
- has widespread use
- has been considerable controversy over its security

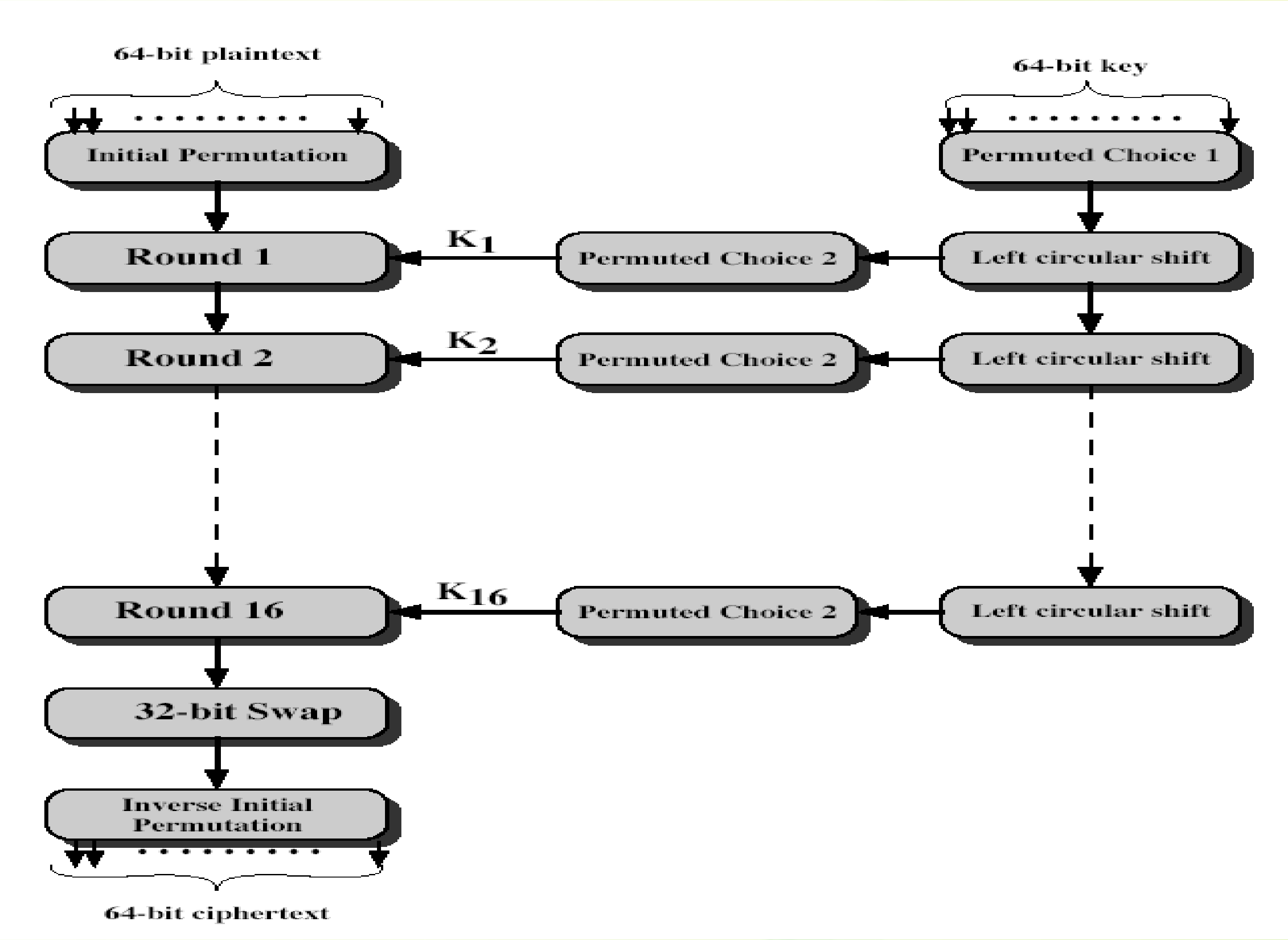
DES History

- 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
- subsequent events and public analysis show in fact design was appropriate
- DES has become widely used, esp in financial applications

DES Encryption



Initial Permutation IP

- first step of the data computation
- IP reorders the input data bits
- even bits to LH half, odd bits to RH half
- quite regular in structure (easy in h/w)
- see text Table 3.2
- example:

IP(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)

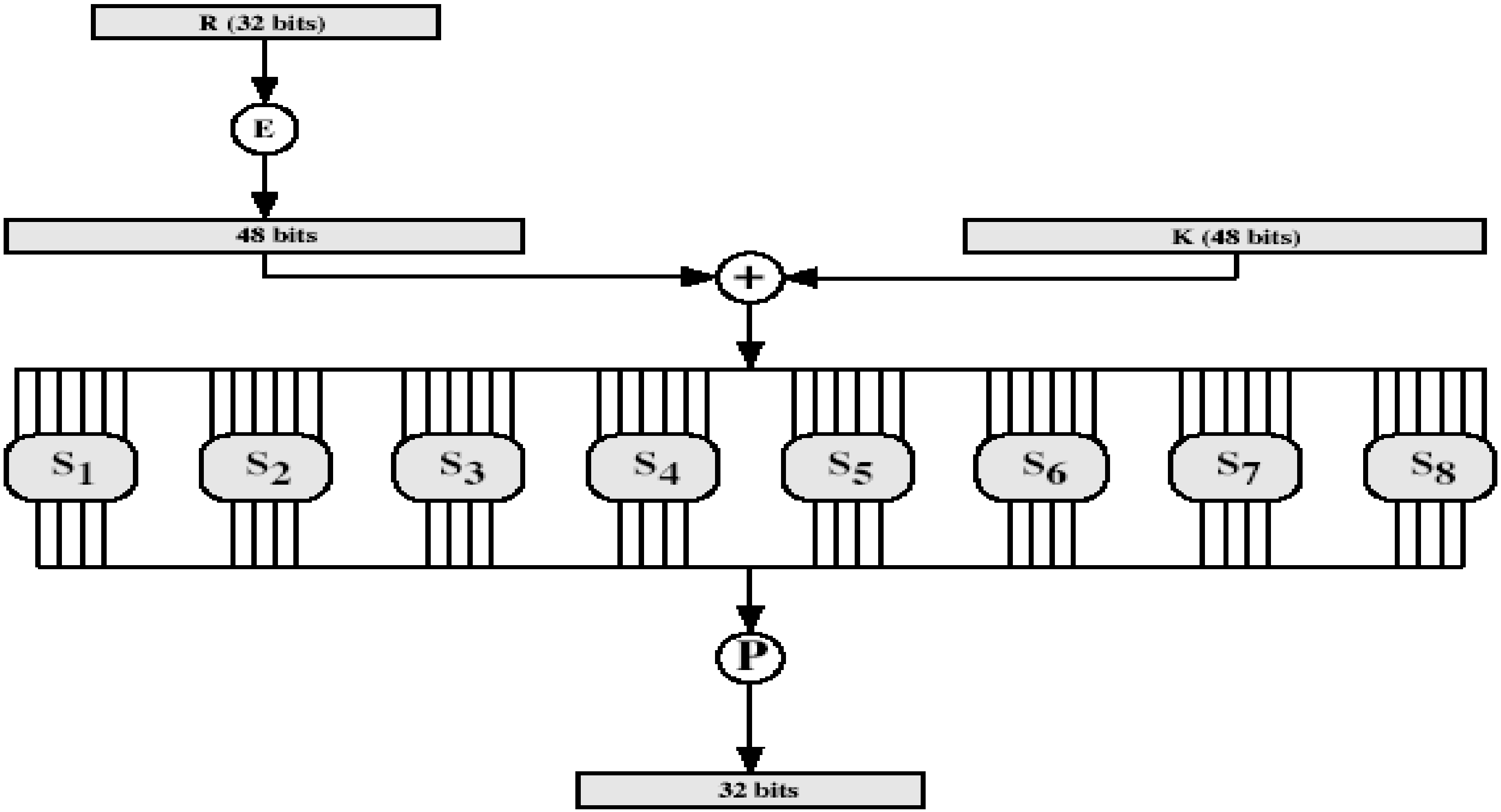
DES Round Structure

- 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
  - passes through 8 S-boxes to get 32-bit result
  - finally permutes this using 32-bit perm P

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) select one rows
  - inner bits 2-5 (**col** bits) are substituted
  - result is 8 lots of 4 bits, or 32 bits
- row selection depends on both data & key
  - feature known as autoclaving (autokeying)
- example:

S(18 09 12 3d 11 17 38 39) = 5fd25e03

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

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

- key desirable property of encryption alg
- where a change of **one** input or key bit results in changing approx **half** output bits
- making 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
  - 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!
- still must be able to recognize plaintext
- now considering alternatives to DES

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 utilise 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
  - related key attacks

Differential Cryptanalysis

- one of the most significant recent (public) advances in cryptanalysis
- known by NSA in 70's cf DES design
- Murphy, Biham & Shamir published 1990
- powerful method to analyse block ciphers
- used to analyse most current block ciphers with varying degrees of success
- DES reasonably resistant to it, cf Lucifer

Differential Cryptanalysis

- a statistical attack against Feistel ciphers
- uses cipher structure not previously used
- design of S-P networks has output of function *f* influenced by both input & key
- hence cannot trace values back through cipher without knowing values of the key
- Differential Cryptanalysis compares two related pairs of encryptions

Differential Cryptanalysis Compares Pairs of Encryptions

- with a known difference in the input
- searching for a known difference in output
- when same subkeys are used

$$\Delta m_{i+1} = m_{i+1} \oplus m'_{i+1}$$

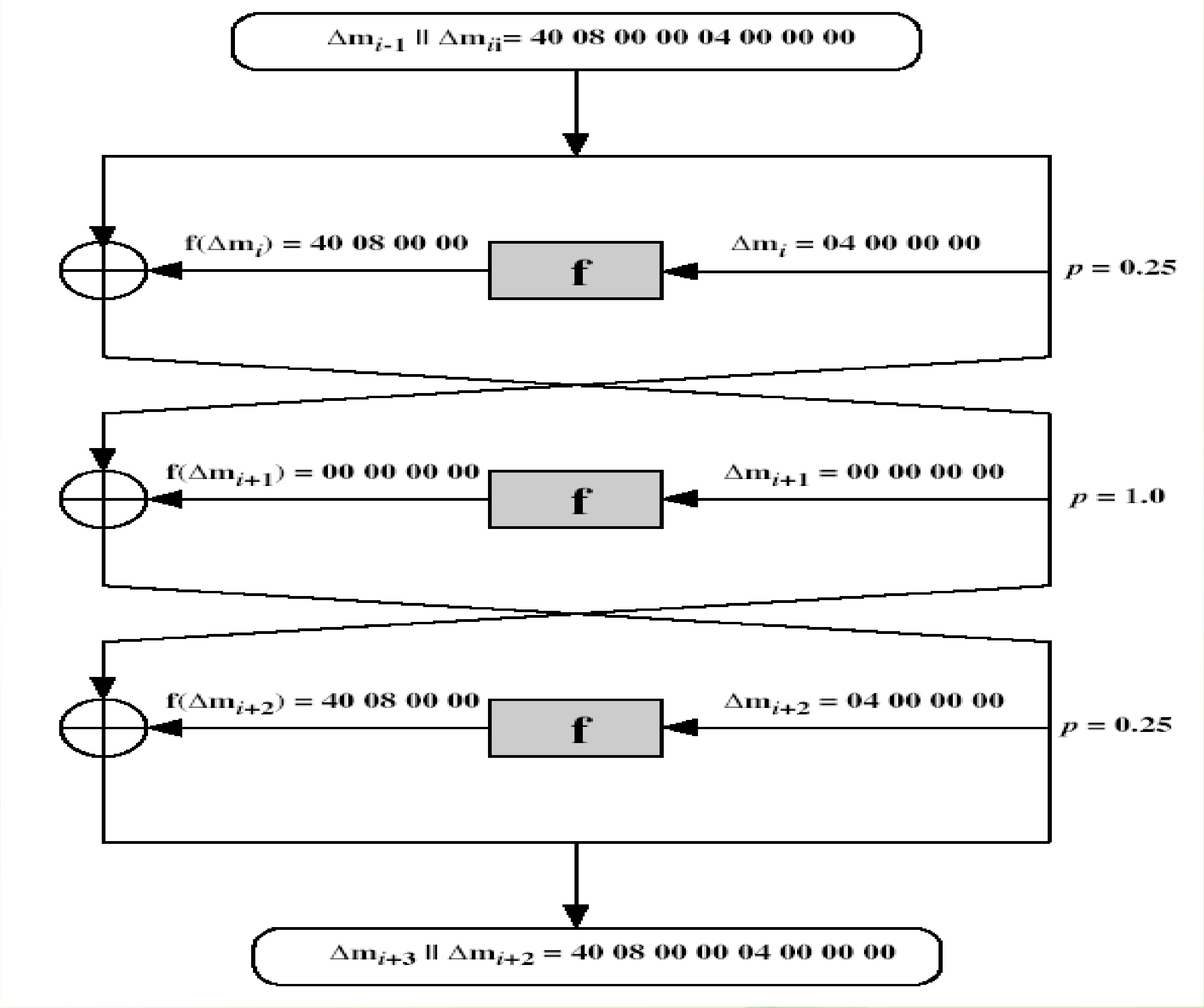
$$= \left[m_{i-1} \oplus f(m_i, K_i)\right] \oplus \left[m'_{i-1} \oplus f(m'_i, K_i)\right]$$

$$= \Delta m_{i-1} \oplus \left[f(m_i, K_i) \oplus f(m'_i, K_i)\right]$$

Differential Cryptanalysis

- have some input difference giving some output difference with probability p
- if find instances of some higher probability input / output difference pairs occurring
- can infer subkey that was used in round
- then must iterate process over many rounds (with decreasing probabilities)

Differential Cryptanalysis



Differential Cryptanalysis

- perform attack by repeatedly encrypting plaintext pairs with known input XOR until obtain desired output XOR
- when found
  - if intermediate rounds match required XOR have a **right pair**
  - if not then have a **wrong pair**, relative ratio is S/N for attack
- can then deduce keys values for the rounds
  - right pairs suggest same key bits
  - wrong pairs give random values
- for large numbers of rounds, probability is so low that more pairs are required than exist with 64-bit inputs
- Biham and Shamir have shown how a 13-round iterated characteristic can break the full 16-round DES

Linear Cryptanalysis

- another recent development
- also a statistical method
- must be iterated over rounds, with decreasing probabilities
- developed by Matsui et al in early 90's
- based on finding linear approximations
- can attack DES with 2^{47} known plaintexts, still in practise infeasible

Linear Cryptanalysis

- find linear approximations with prob p != ½

  P[i1,i2,...,ia](+)C[j1,j2,...,jb] = K[k1,k2,...,kc]

  where ia,jb,kc are bit locations in P,C,K
- gives linear equation for key bits
- get one key bit using max likelihood alg
- using a large number of trial encryptions
- effectiveness given by: |p–½|

Block Cipher Design Principles

- basic principles still like Feistel in 1970's
- number of rounds
  - more is better, exhaustive search best attack
- function f:
  - provides "confusion", is nonlinear, avalanche
- key schedule
  - complex subkey creation, key avalanche

Modes of Operation

- block ciphers encrypt fixed size blocks
- eg. DES encrypts 64-bit blocks, with 56-bit key
- need way to use in practise, given usually have arbitrary amount of information to encrypt
- four were defined for DES in ANSI standard **ANSI X3.106-1983 Modes of Use**
- subsequently now have 5 for DES and AES
- have **block** and **stream** modes

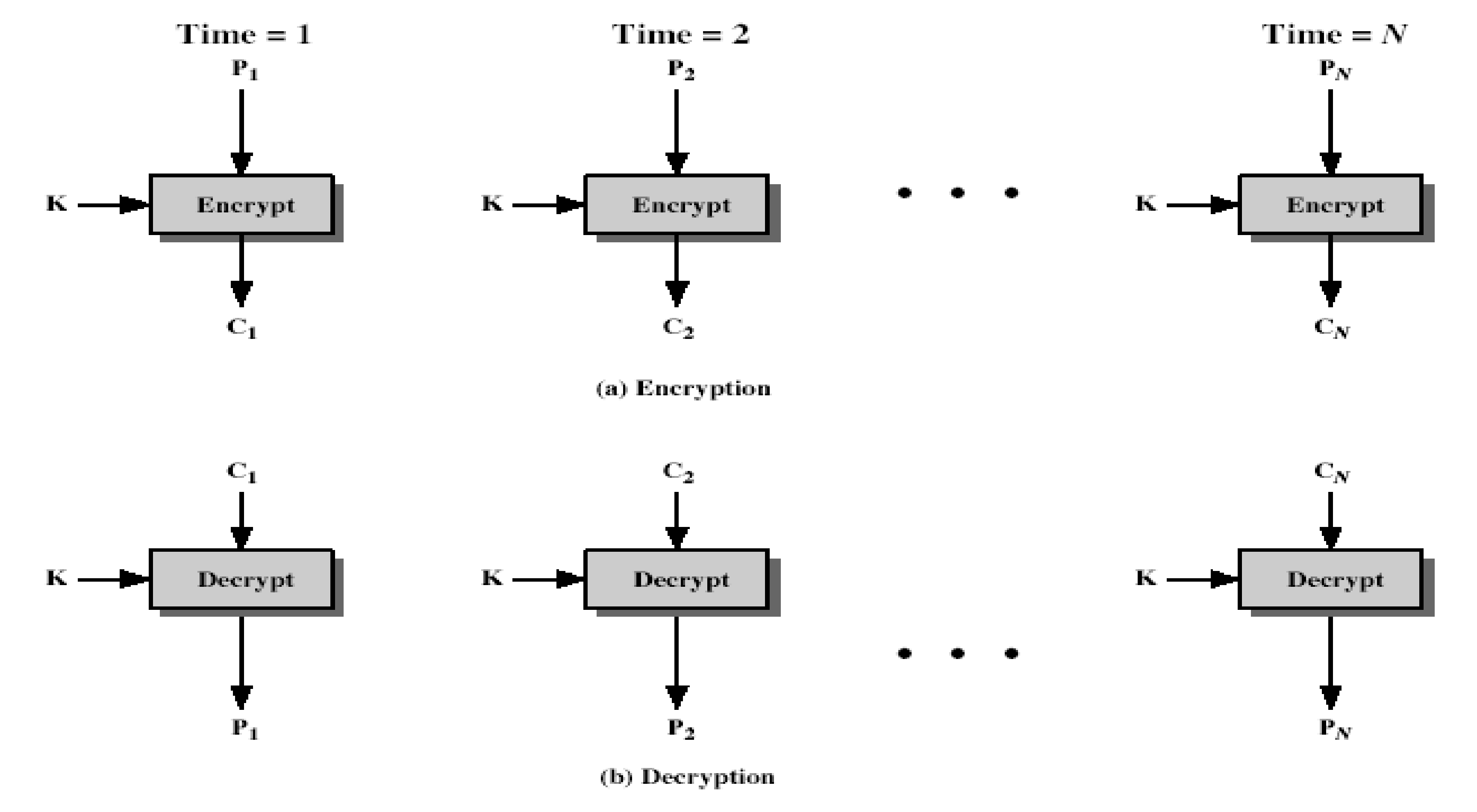
Electronic Codebook Book (ECB)

- message is broken into independent blocks which are encrypted
- each block is a value which is substituted, like a codebook, hence name
- each block is encoded independently of the other blocks

  $C_i = DES_{K1} (P_i)$

- uses: secure transmission of single values

Electronic Codebook Book (ECB)



Advantages and Limitations of ECB

- repetitions in message may show in ciphertext
  - if aligned with message block
  - particularly with data such graphics
  - or with messages that change very little, which become a code-book analysis problem
- weakness due to encrypted message blocks being independent
- main use is sending a few blocks of data

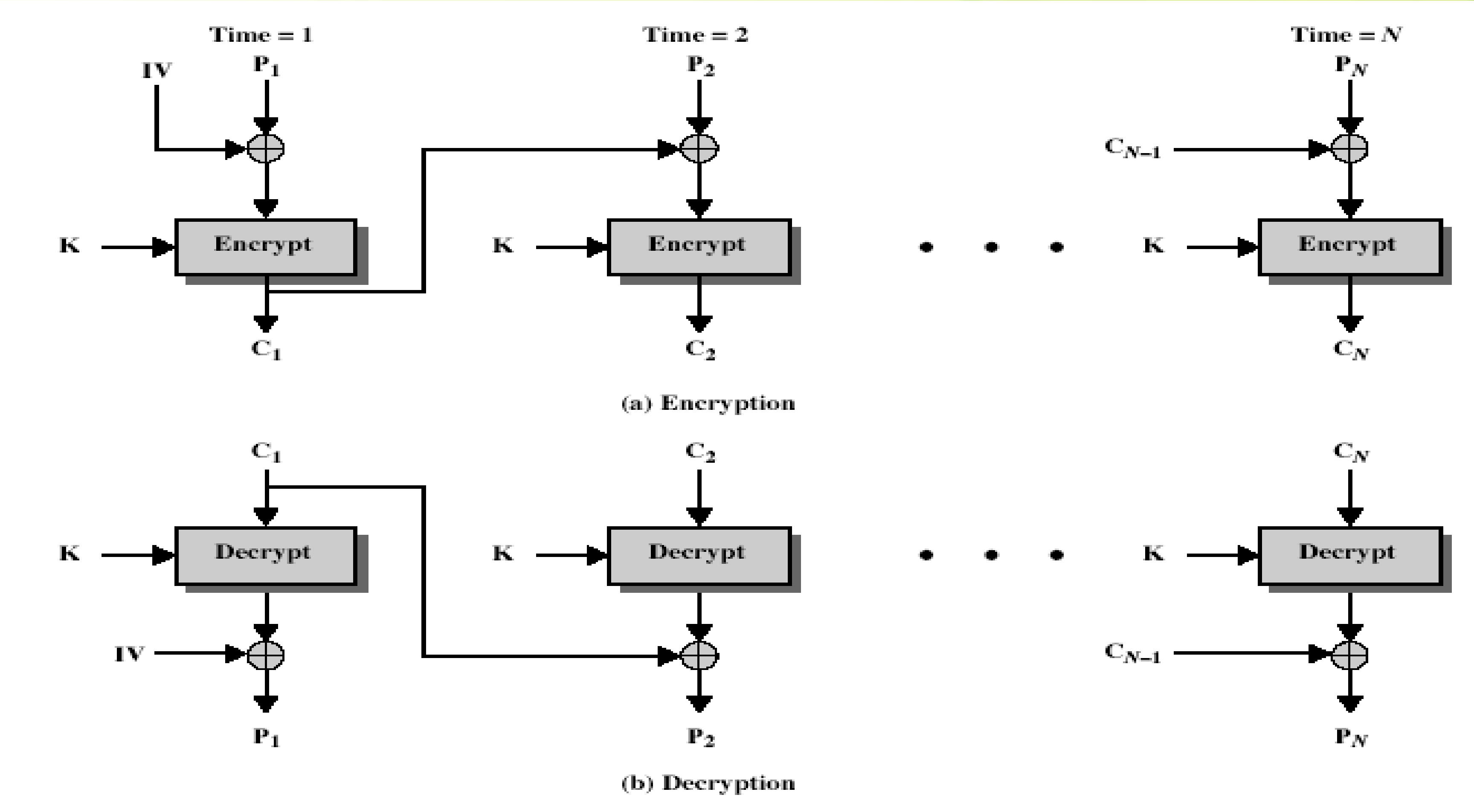
Cipher Block Chaining (CBC)

- 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

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

  $C_{-1} = IV$
- uses: bulk data encryption, authentication

Cipher Block Chaining (CBC)



Advantages and Limitations of CBC

- 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 (as in EFTPOS) or it must be sent encrypted in ECB mode before rest of message
- at end of message, handle possible last short block
  - by padding either with known non-data value (eg nulls)
  - or pad last block with count of pad size
    - eg. [b1 b2 b3 0 0 0 0 5] <- 3 data bytes, then 5 bytes pad+count

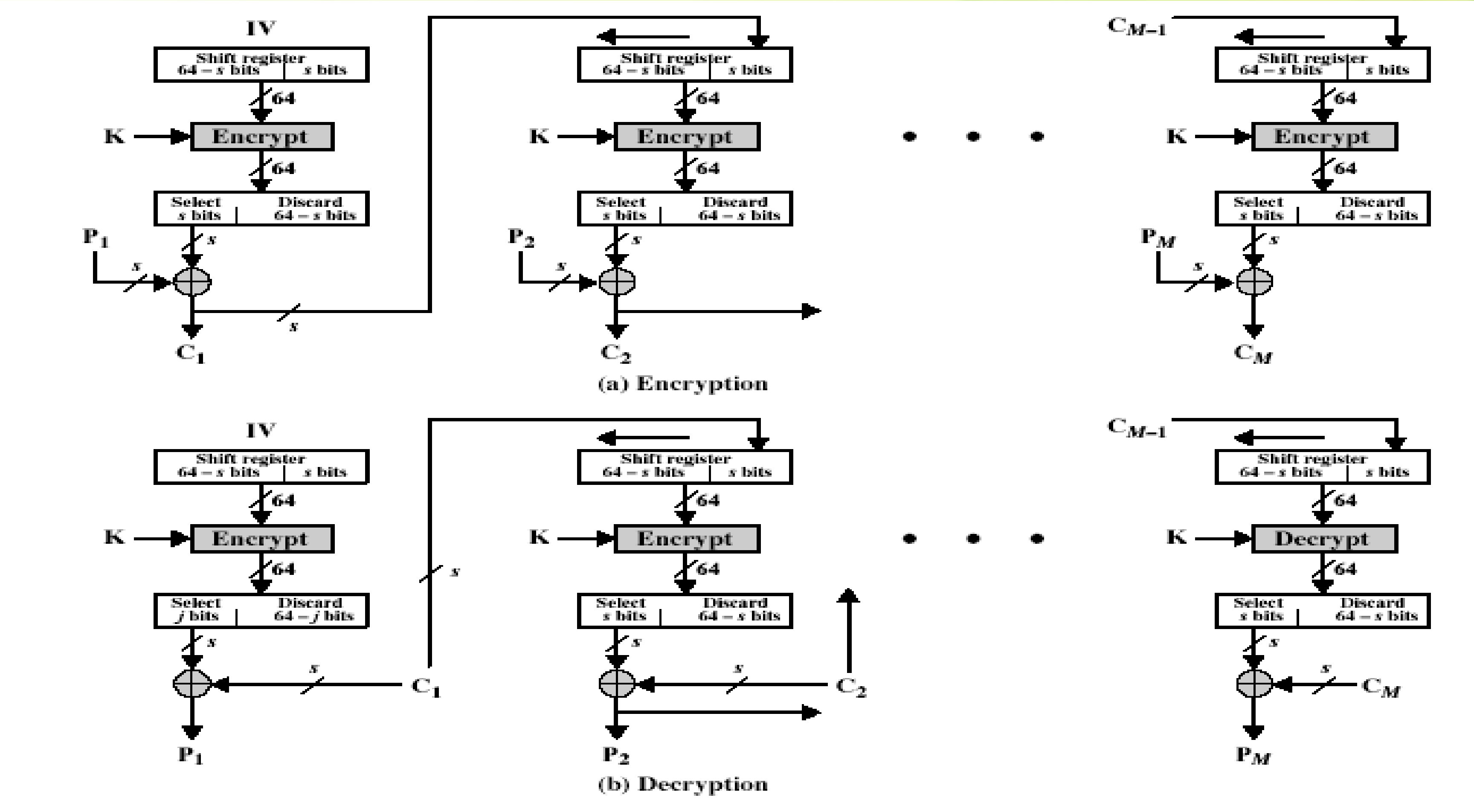
Cipher Feedback (CFB)

- message is treated as a stream of bits
- added to the output of the block cipher
- result is feed back for next stage (hence name)
- standard allows any number of bit (1,8 or 64 or whatever) to be feed back
  - denoted CFB-1, CFB-8, CFB-64 etc
- is most efficient to use all 64 bits (CFB-64)

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

  $C_{-1} = IV$
- uses: stream data encryption, authentication

Cipher Feedback (CFB)



Advantages and Limitations of CFB

- appropriate when data arrives in bits/bytes
- most common stream mode
- limitation is need to stall while do block encryption after every n-bits
- note that the block cipher is used in **encryption** mode at **both** ends
- errors propogate for several blocks after the error

Output Feedback (OFB)

- message is treated as a stream of bits
- output of cipher is added to message
- output is then feed back (hence name)
- feedback is independent of message
- can be computed in advance

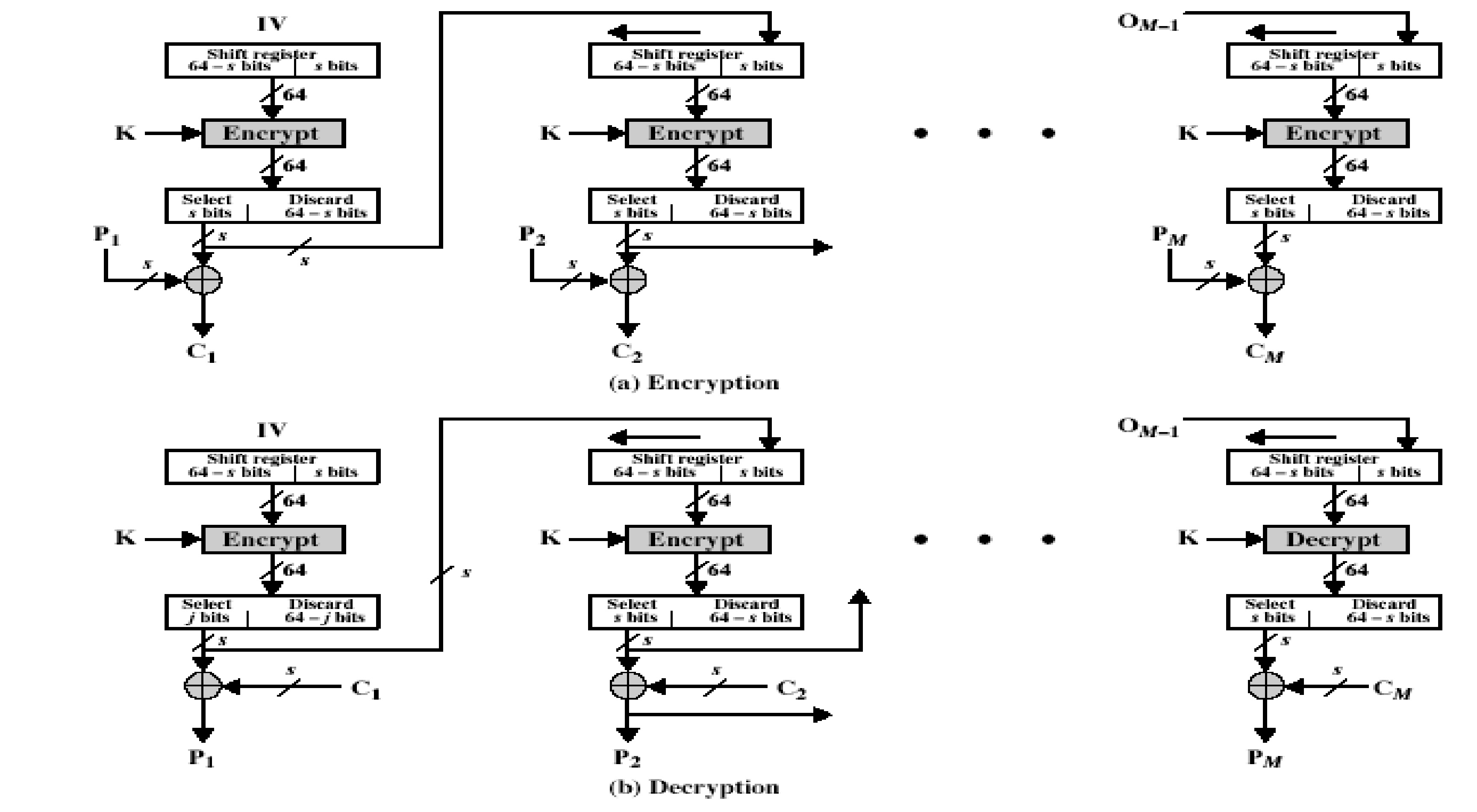
  $C_i = P_i \text{ XOR } O_i$

  $O_i = DES_{K1}(O_{i-1})$

  $O_{-1} = IV$

- uses: stream encryption over noisy channels

Output Feedback (OFB)



Advantages and Limitations of OFB

- used when error feedback a problem or where need to encryptions before message is available
- superficially similar to CFB
- but feedback is from the output of cipher and is independent of message
- a variation of a Vernam cipher
  - hence must **never** reuse the same sequence (key+IV)
- sender and receiver must remain in sync, and some recovery method is needed to ensure this occurs
- originally specified with m-bit feedback in the standards
- subsequent research has shown that only **OFB-64** should ever be used

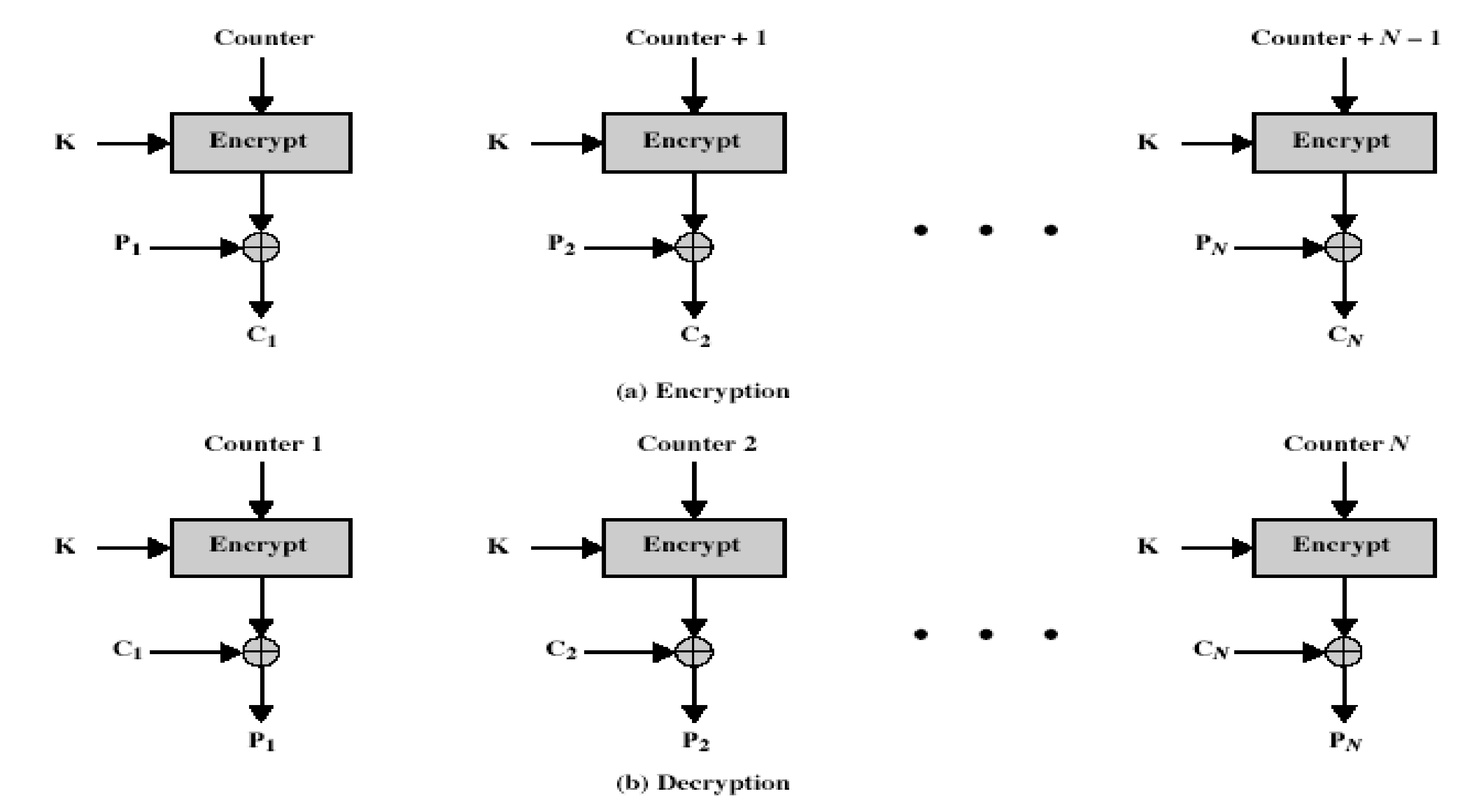
Counter (CTR)

- a “new” mode, though proposed early on
- similar to OFB but encrypts counter value rather than any feedback value
- must have a different key & counter value for every plaintext block (never reused)

  $C_i = P_i \text{ XOR } O_i$

  $O_i = DES_{K1}(i)$
- uses: high-speed network encryptions

Counter (CTR)


Counter
Counter + 1
Counter + N – 1
K
Encrypt
P1
P2
PN
C1
C2
CN
(a) Encryption
Counter 1
Counter 2
Counter N
K
Encrypt
C1
C2
CN
P1
P2
PN
(b) Decryption

Advantages and Limitations of CTR

- efficiency
  - can do parallel encryptions
  - in advance of need
  - good for bursty high speed links
- random access to encrypted data blocks
- provable security (good as other modes)
- but must ensure never reuse key/counter values, otherwise could break (cf OFB)

Summary

- have considered:
- block cipher design principles
- DES
  - details
  - strength
- Differential & Linear Cryptanalysis
- Modes of Operation
  - ECB, CBC, CFB, OFB, CTR

CRYPTOGRAPHY AND NETWORK SECURITY

Unit-02 (Continued…)

Advanced Encryption Standard

Advanced Encryption Standard

"It seems very simple."

"It is very simple. But if you don't know what the key is it's virtually indecipherable."

—*Talking to Strange Men,* Ruth Rendell

Origins

- clear a replacement for DES was needed
  - have theoretical attacks that can break it
  - have demonstrated exhaustive key search attacks
- can use Triple-DES – but slow, has small blocks
- US NIST issued call for ciphers in 1997
- 15 candidates accepted in Jun 98
- 5 were shortlisted in Aug-99
- Rijndael was selected as the AES in Oct-2000
- issued as FIPS PUB 197 standard in Nov-2001

AES Requirements

- private key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- stronger & faster than Triple-DES
- active life of 20-30 years (+ archival use)
- provide full specification & design details
- both C & Java implementations
- NIST have released all submissions & unclassified analyses

AES Evaluation Criteria

- initial criteria:
  - security – effort for practical cryptanalysis
  - cost – in terms of computational efficiency
  - algorithm & implementation characteristics
- final criteria
  - general security
  - ease of software & hardware implementation
  - implementation attacks
  - flexibility (in en/decrypt, keying, other factors)

AES Shortlist

- after testing and evaluation, shortlist in Aug-99:
  - MARS (IBM) - complex, fast, high security margin
  - RC6 (USA) - v. simple, v. fast, low security margin
  - Rijndael (Belgium) - clean, fast, good security margin
  - Serpent (Euro) - slow, clean, v. high security margin
  - Twofish (USA) - complex, v. fast, high security margin
- then subject to further analysis & comment
- saw contrast between algorithms with
  - few complex rounds verses many simple rounds
  - which refined existing ciphers verses new proposals

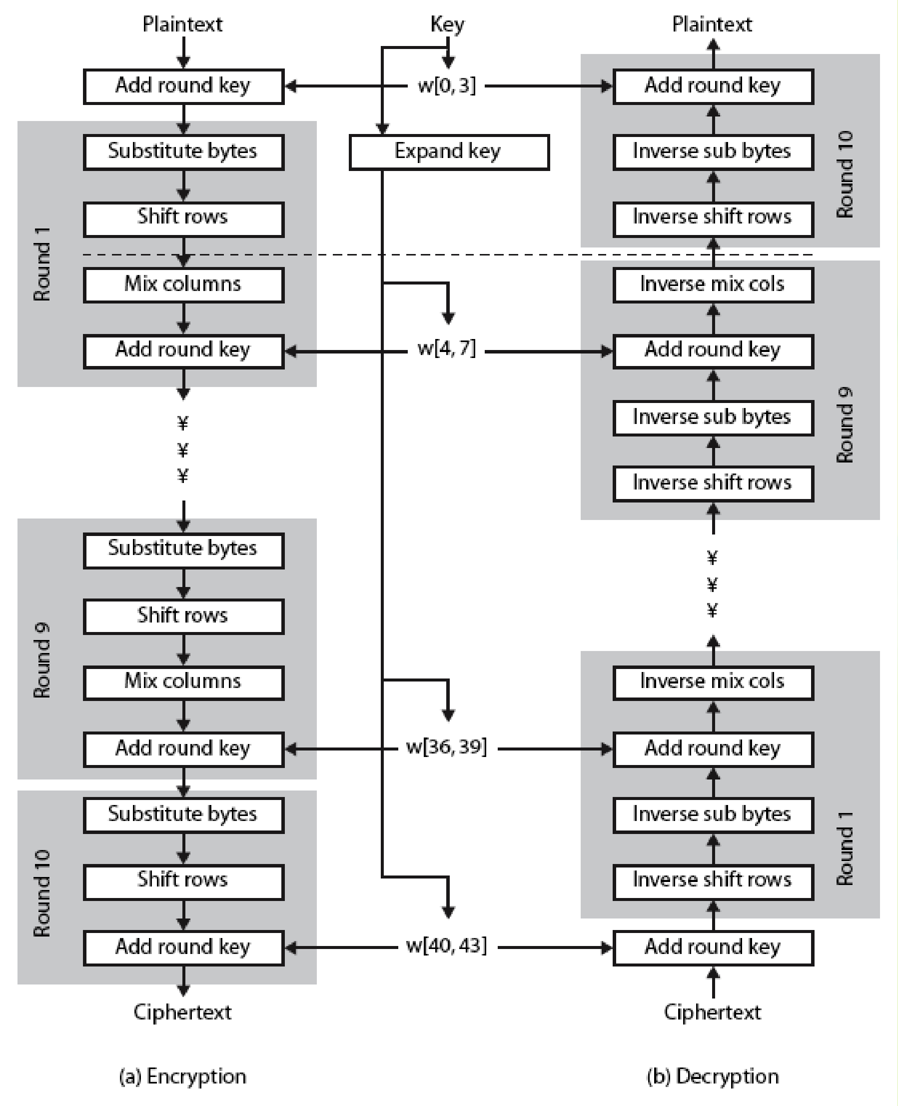
The AES Cipher - Rijndael

- designed by Rijmen-Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data
- an **iterative** rather than **feistel** cipher
  - processes data as block of 4 columns of 4 bytes
  - operates on entire data block in every round
- designed to be:
  - resistant against known attacks
  - speed and code compactness on many CPUs
  - design simplicity

Rijndael

- data block of 4 columns of 4 bytes is state
- key is expanded to array of words
- has 9/11/13 rounds in which state undergoes:
  - byte substitution (1 S-box used on every byte)
  - shift rows (permute bytes between groups/columns)
  - mix columns (subs using matrix multipy of groups)
  - add round key (XOR state with key material)
  - view as alternating XOR key & scramble data bytes
- initial XOR key material & incomplete last round
- with fast XOR & table lookup implementation

Rijndael

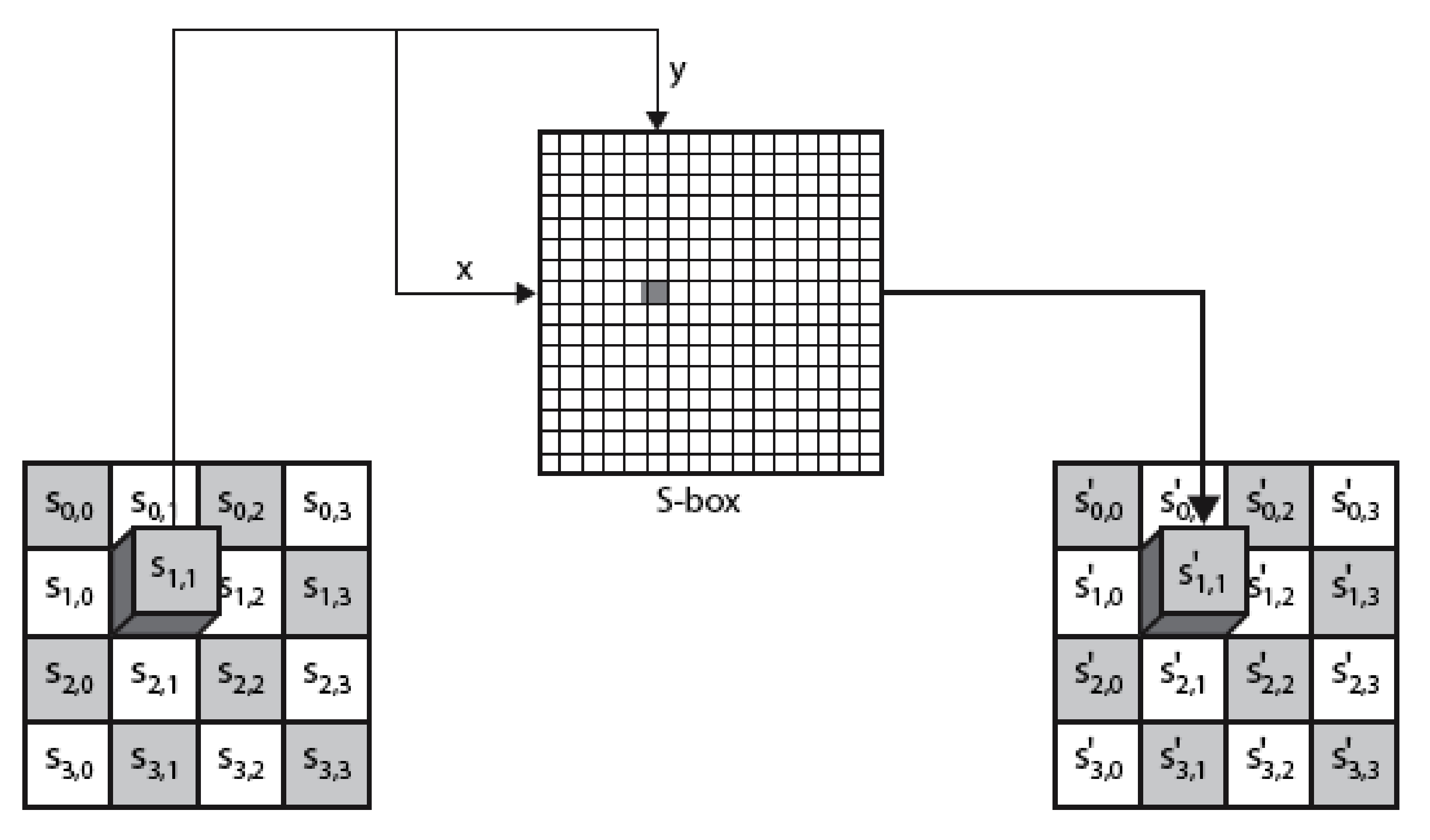


Plaintext
Key
Plaintext
Add round key
w[0, 3]
Add round key
Substitute bytes
Expand key
Inverse sub bytes
Round 10
Shift rows
Inverse shift rows
Round 1
Mix columns
Inverse mix cols
Add round key
w[4, 7]
Add round key
Inverse sub bytes
Round 9
Inverse shift rows
¥
¥
¥
¥
¥
¥
Substitute bytes
Inverse mix cols
Shift rows
Round 9
Mix columns
Add round key
w[36, 39]
Add round key
Inverse sub bytes
Round 1
Substitute bytes
Inverse shift rows
Shift rows
Round 10
Add round key
w[40, 43]
Add round key
Ciphertext
Ciphertext
(a) Encryption
(b) Decryption

Byte Substitution

- a simple substitution of each byte
- uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
- each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)
  - eg. byte {95} is replaced by byte in row 9 column 5
  - which has value {2A}
- S-box constructed using defined transformation of values in $GF(2^8)$
- designed to be resistant to all known attacks

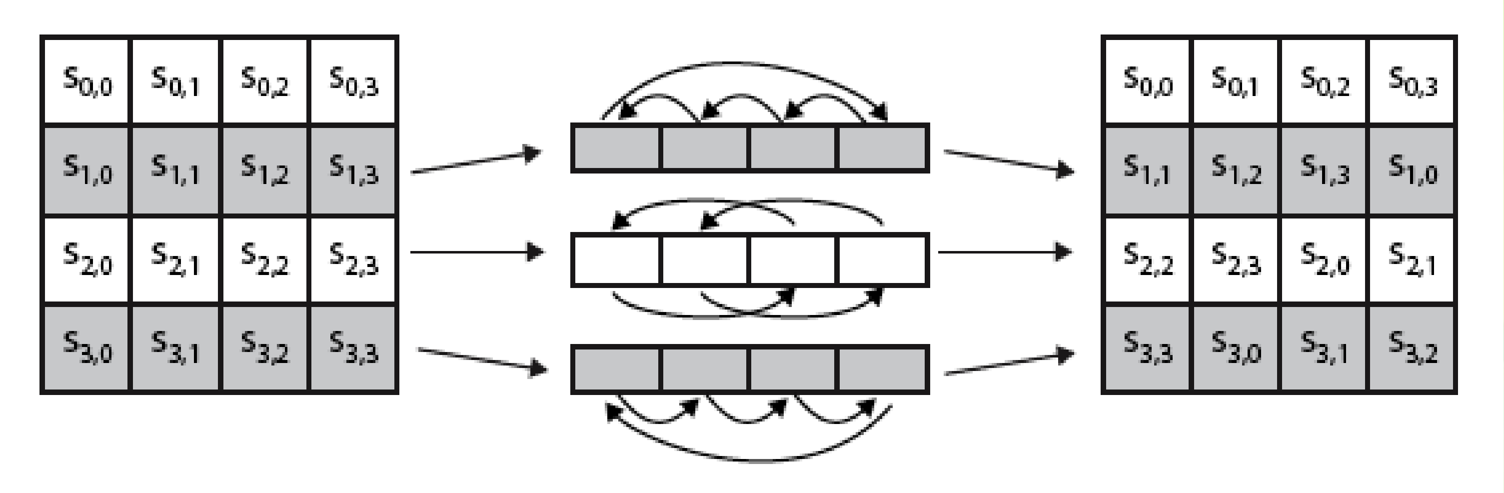
Byte Substitution



Shift Rows

- a circular byte shift in each each
  - 1st row is unchanged
  - 2nd row does 1 byte circular shift to left
  - 3rd row does 2 byte circular shift to left
  - 4th row does 3 byte circular shift to left
- decrypt inverts using shifts to right
- since state is processed by columns, this step permutes bytes between the columns

Shift Rows



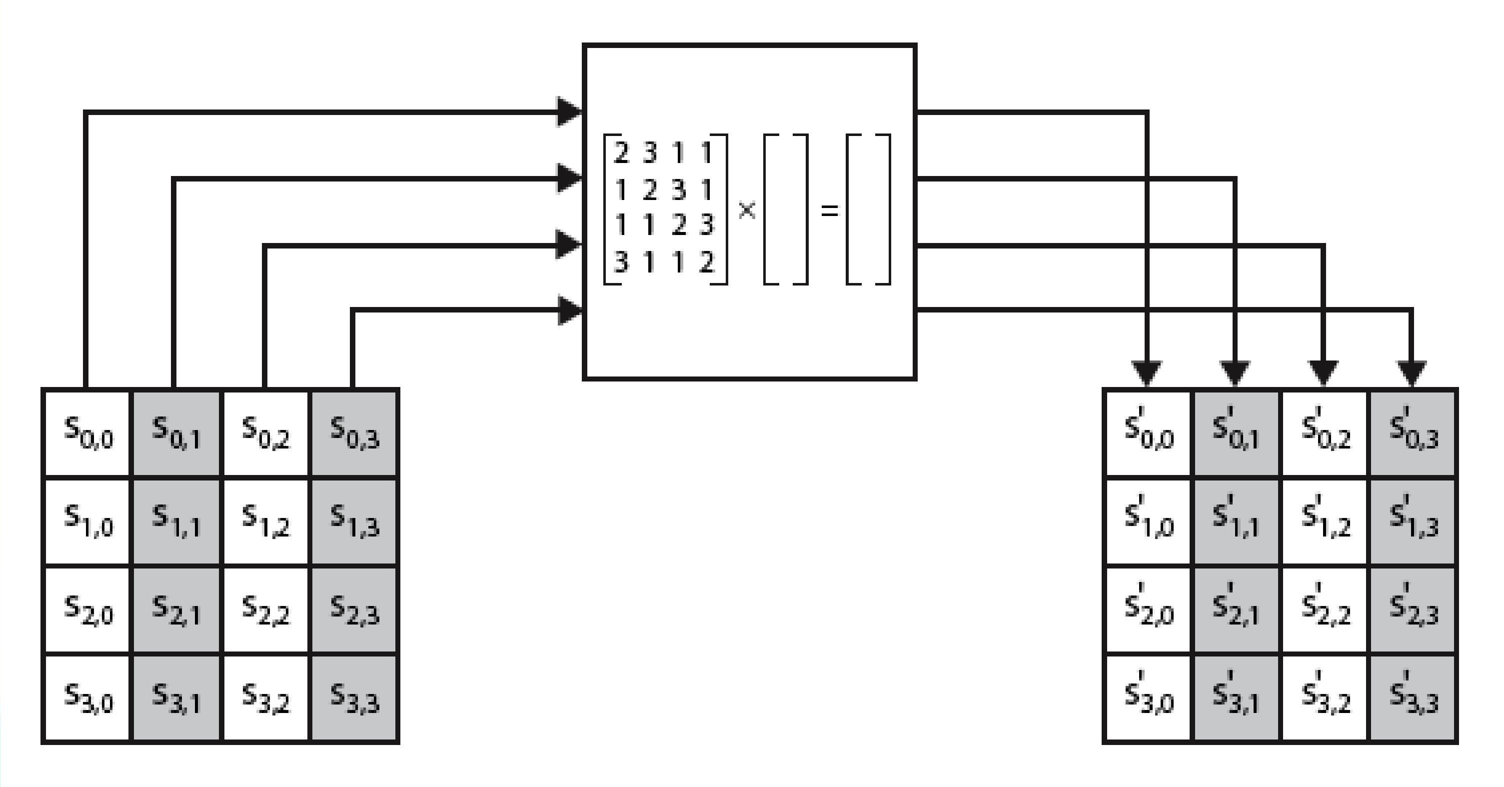
s0,0 s0,1 s0,2 s0,3
s1,0 s1,1 s1,2 s1,3
s2,0 s2,1 s2,2 s2,3
s3,0 s3,1 s3,2 s3,3
s0,0 s0,1 s0,2 s0,3
s1,1 s1,2 s1,3 s1,0
s2,2 s2,3 s2,0 s2,1
s3,3 s3,0 s3,1 s3,2

Mix Columns

- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in GF(2^8) using prime poly $m(x) = x^8+x^4+x^3+x+1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Mix Columns



2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
s0,0 s0,1 s0,2 s0,3
s1,0 s1,1 s1,2 s1,3
s2,0 s2,1 s2,2 s2,3
s3,0 s3,1 s3,2 s3,3
s'0,0 s'0,1 s'0,2 s'0,3
s'1,0 s'1,1 s'1,2 s'1,3
s'2,0 s'2,1 s'2,2 s'2,3
s'3,0 s'3,1 s'3,2 s'3,3

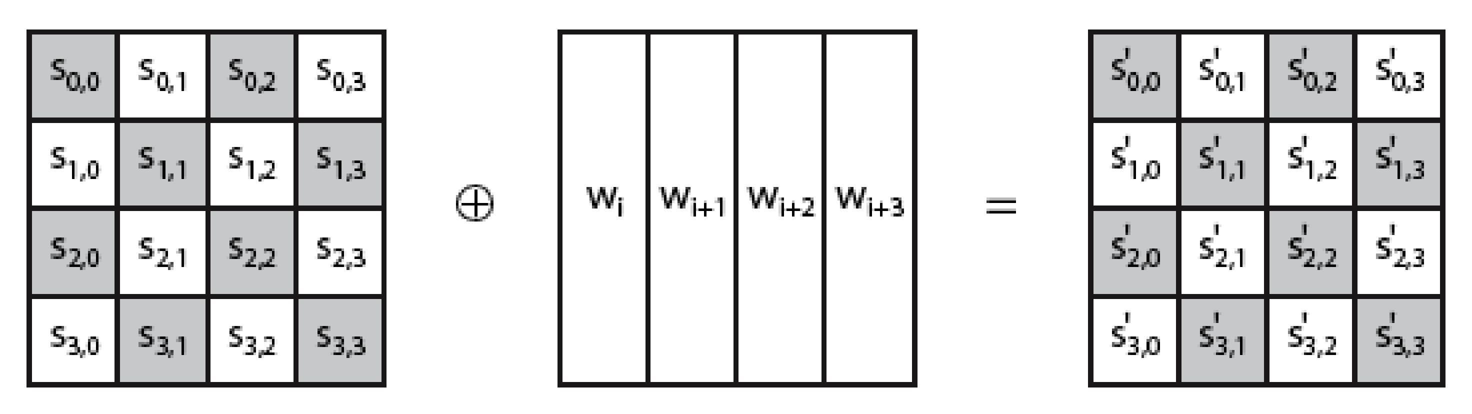
Mix Columns

- can express each col as 4 equations
  - to derive each new byte in col
- decryption requires use of inverse matrix
  - with larger coefficients, hence a little harder
- have an alternate characterisation
  - each column a 4-term polynomial
  - with coefficients in $GF(2^8)$
  - and polynomials multiplied modulo (x^4+1)

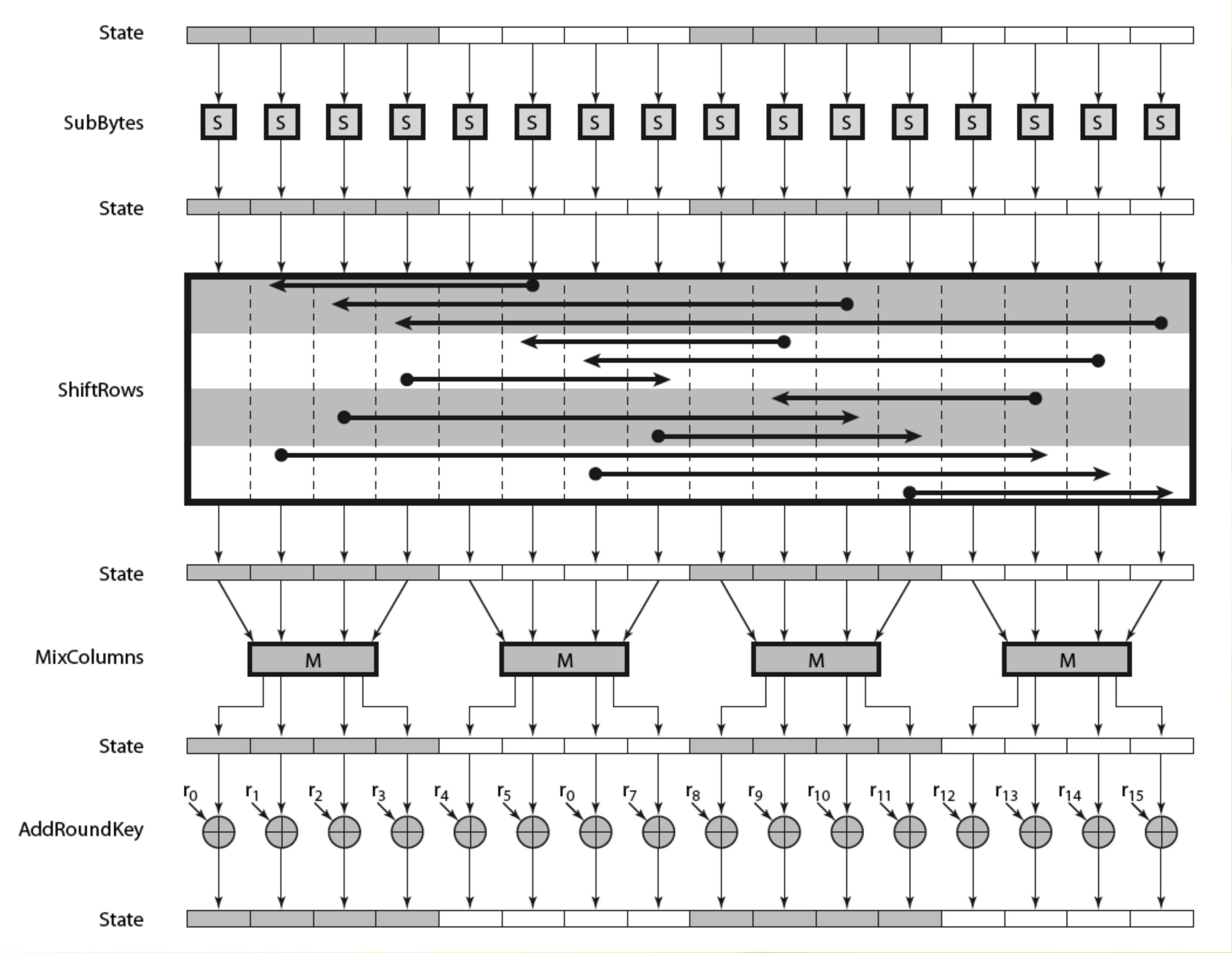
Add Round Key

- XOR state with 128-bits of the round key
- again processed by column (though effectively a series of byte operations)
- inverse for decryption identical
  - since XOR own inverse, with reversed keys
- designed to be as simple as possible
  - a form of Vernam cipher on expanded key
  - requires other stages for complexity / security

Add Round Key



AES Round

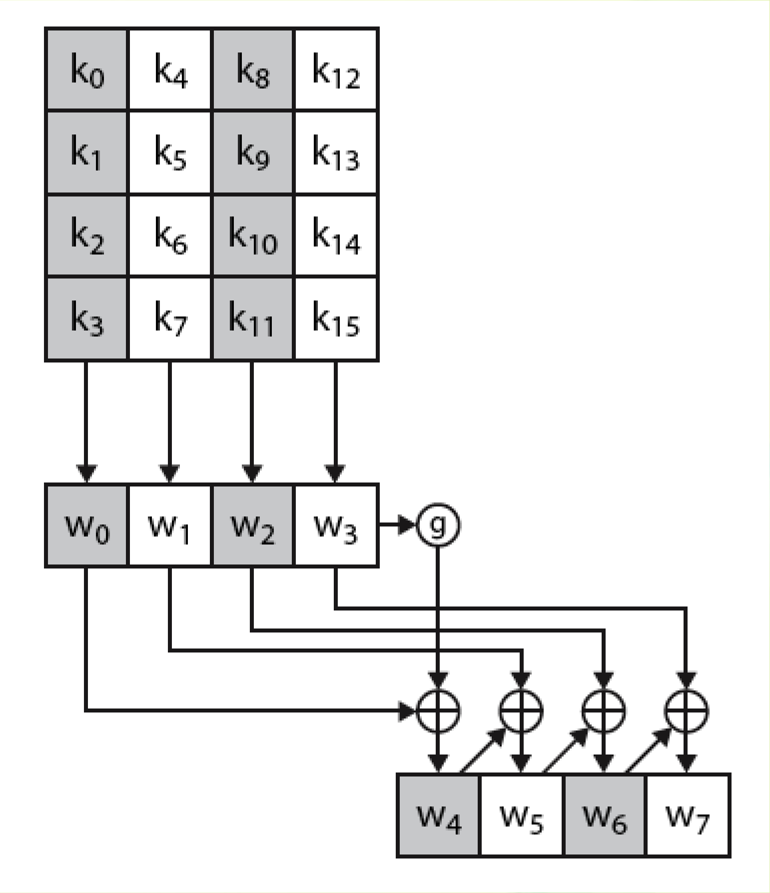


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r_0 r_1 r_2 r_3 r_4 r_5 r_0 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}
State

AES Key Expansion

- takes 128-bit (16-byte) key and expands into array of 44/52/60 32-bit words
- start by copying key into first 4 words
- then loop creating words that depend on values in previous & 4 places back
  - in 3 of 4 cases just XOR these together
  - 1st word in 4 has rotate + S-box + XOR round constant on previous, before XOR 4th back

AES Key Expansion



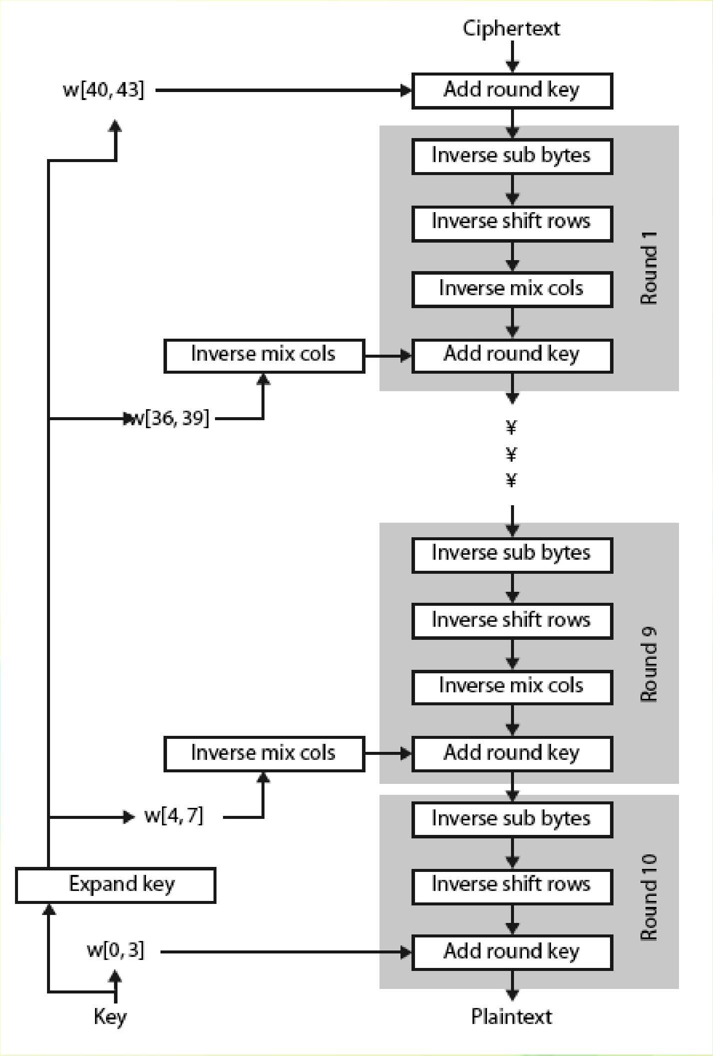
Key Expansion Rationale

- designed to resist known attacks
- design criteria included
  - knowing part key insufficient to find many more
  - invertible transformation
  - fast on wide range of CPU's
  - use round constants to break symmetry
  - diffuse key bits into round keys
  - enough non-linearity to hinder analysis
  - simplicity of description

AES Decryption

- AES decryption is not identical to encryption since steps done in reverse
- but can define an equivalent inverse cipher with steps as for encryption
  - but using inverses of each step
  - with a different key schedule
- works since result is unchanged when
  - swap byte substitution & shift rows
  - swap mix columns & add (tweaked) round key

AES Decryption



Implementation Aspects

- can efficiently implement on 8-bit CPU
  - byte substitution works on bytes using a table of 256 entries
  - shift rows is simple byte shift
  - add round key works on byte XOR's
  - mix columns requires matrix multiply in $GF(2^8)$ which works on byte values, can be simplified to use table lookups & byte XOR's

Implementation Aspects

- can efficiently implement on 32-bit CPU
  - redefine steps to use 32-bit words
  - can pre-compute 4 tables of 256-words
  - then each column in each round can be computed using 4 table lookups + 4 XORs
  - at a cost of 4Kb to store tables
- designers believe this very efficient implementation was a key factor in its selection as the AES cipher

Summary

- have considered:
  - the AES selection process
  - the details of Rijndael – the AES cipher
  - looked at the steps in each round
  - the key expansion
  - implementation aspects

Queries ...?