# INFORMATION SECURITY

**Course code:ACS013**

**IV. B.Tech II semester**

**(Regulation: IARE R-16)**


BY

Ms Geetavani B

Assistant Professors

Ms B Swathi, Ms. B Anupama, Ms. P Navya

**DEPARTMENT OF COMPUTER SCIENCE OF ENGINEERING**

**INSTITUTE OF AERONAUTICAL ENGINEERING**

**(Autonomous)**

**DUNDIGAL, HYDERABAD - 500 043**

| COs | Course Outcomes |
| --- | --- |
| CO1 | Understand the basic concepts of attacks on computer, computer security. |
| CO2 | Understand the concepts of symmetric key ciphers. |
| CO3 | Describe the message authentication algorithm and hash functions. |
| CO4 | Understand the concepts of e-mail security. |

| CO's | Course outcomes |
| --- | --- |
| CO5 | Understand the concepts of web security. |


# UNIT– I
# ATTACKS ON COMPUTERS AND COMPUTER SECURITY

| CLOs | Course Learning Outcome |
|---|---|
| CLO1 | Understand the different types of attacks, security mechanisms, security services. |
| CLO2 | Explain various substitution techniques such as play-fair cipher, mono-alphabetic cipher and hill cipher. |
| CLO3 | Understand various Transposition techniques such as row transposition and rail-fence. |
| CLO4 | Describe the role of private and public key in encryption and decryption and key size. |

| CLOs | Course Learning Outcome |
| --- | --- |
| CLO5 | Apply the symmetric algorithm for message transmission and analyze the security level of it. |
| CLO6 | Understand various asymmetric key encryption algorithms for message encryption and decryption. |

# Definitions

- **Computer Security** - generic name for the collection of tools designed to protect data and to thwart hackers
- **Network Security** - measures to protect data during their transmission
- **Internet Security** - measures to protect data during their transmission over a collection of interconnected networks

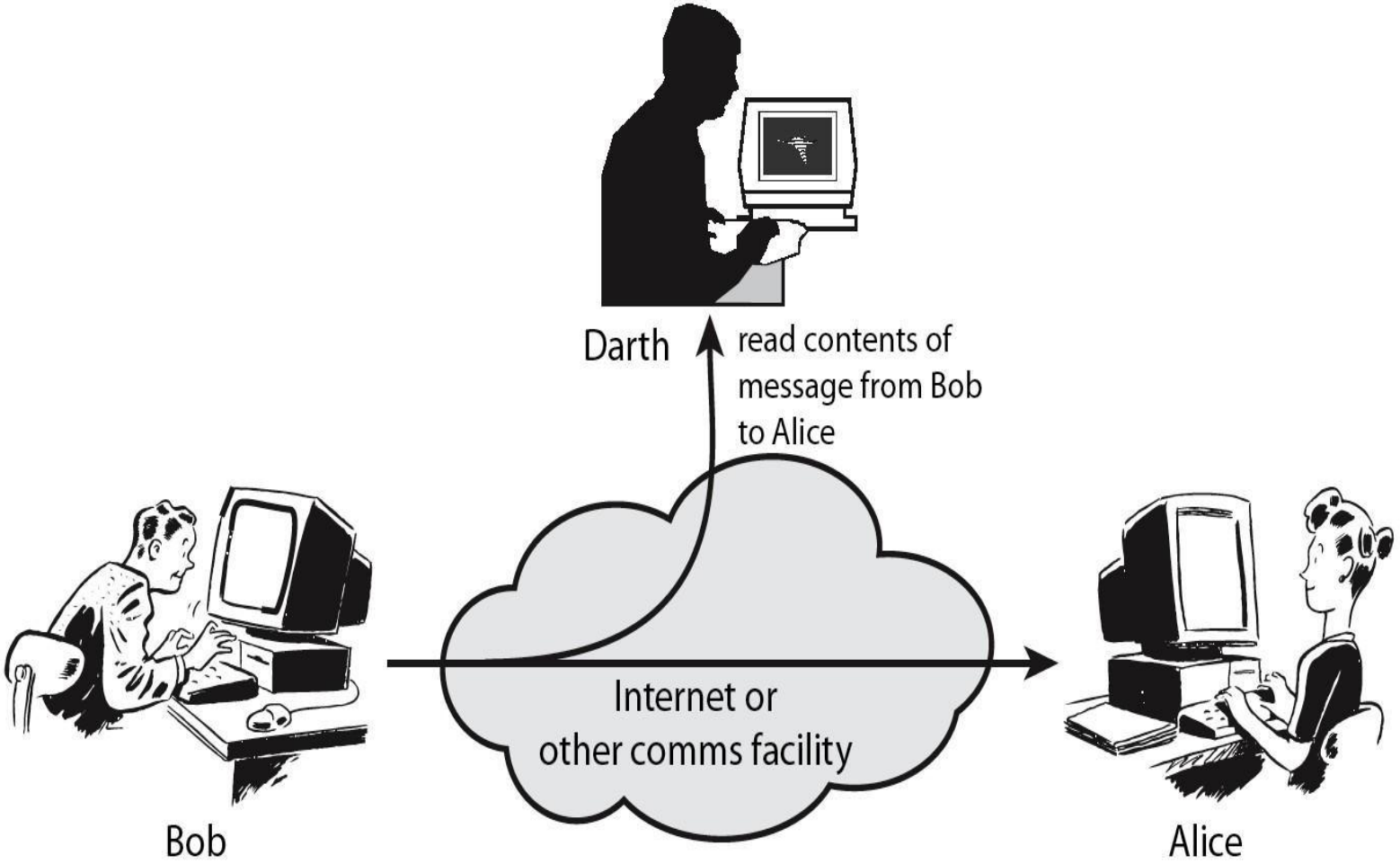
# Aspects of Security

- Consider 3 aspects of information security:
  - Security attack
  - Security mechanism
  - Security service

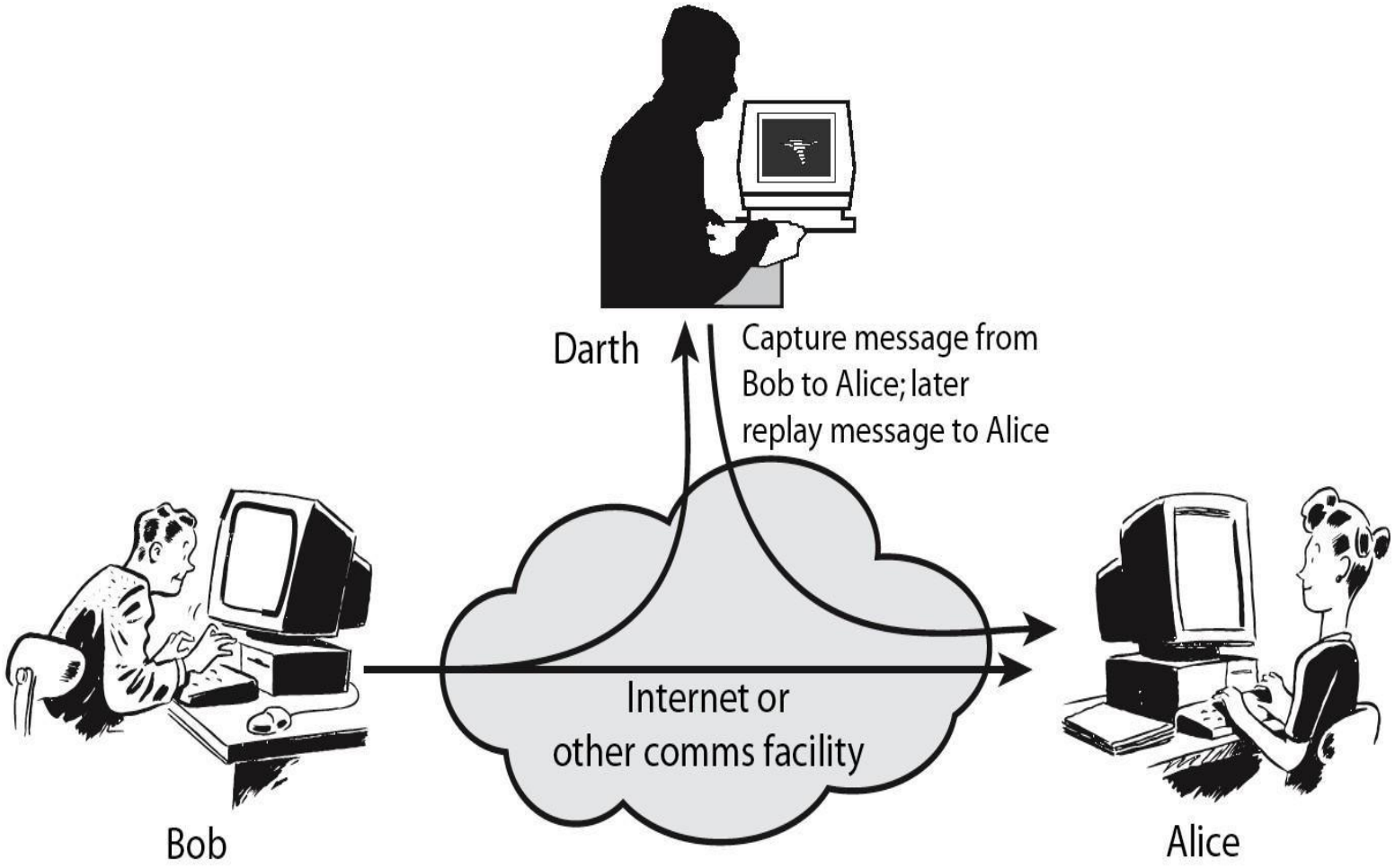
# Security Attack

- Any action that compromises the security of information owned by an organization
- Information security is about how to prevent attacks, or failing that, to detect attacks on information-based systems
- Often *threat* & *attack* used to mean same thing
- Have a wide range of attacks
- Can focus of generic types of attacks
  - Passive
  - Active

# Passive Attacks



# Active Attacks



# Security Services (X.800)

- **Authentication** - assurance that the communicating entity is the one claimed
- **Access Control** - prevention of the unauthorized use of a resource
- **Data Confidentiality** – protection of data from unauthorized disclosure
- **Data Integrity** - assurance that data received is as sent by an authorized entity
- **Non-Repudiation** - protection against denial by one of the parties in a communication

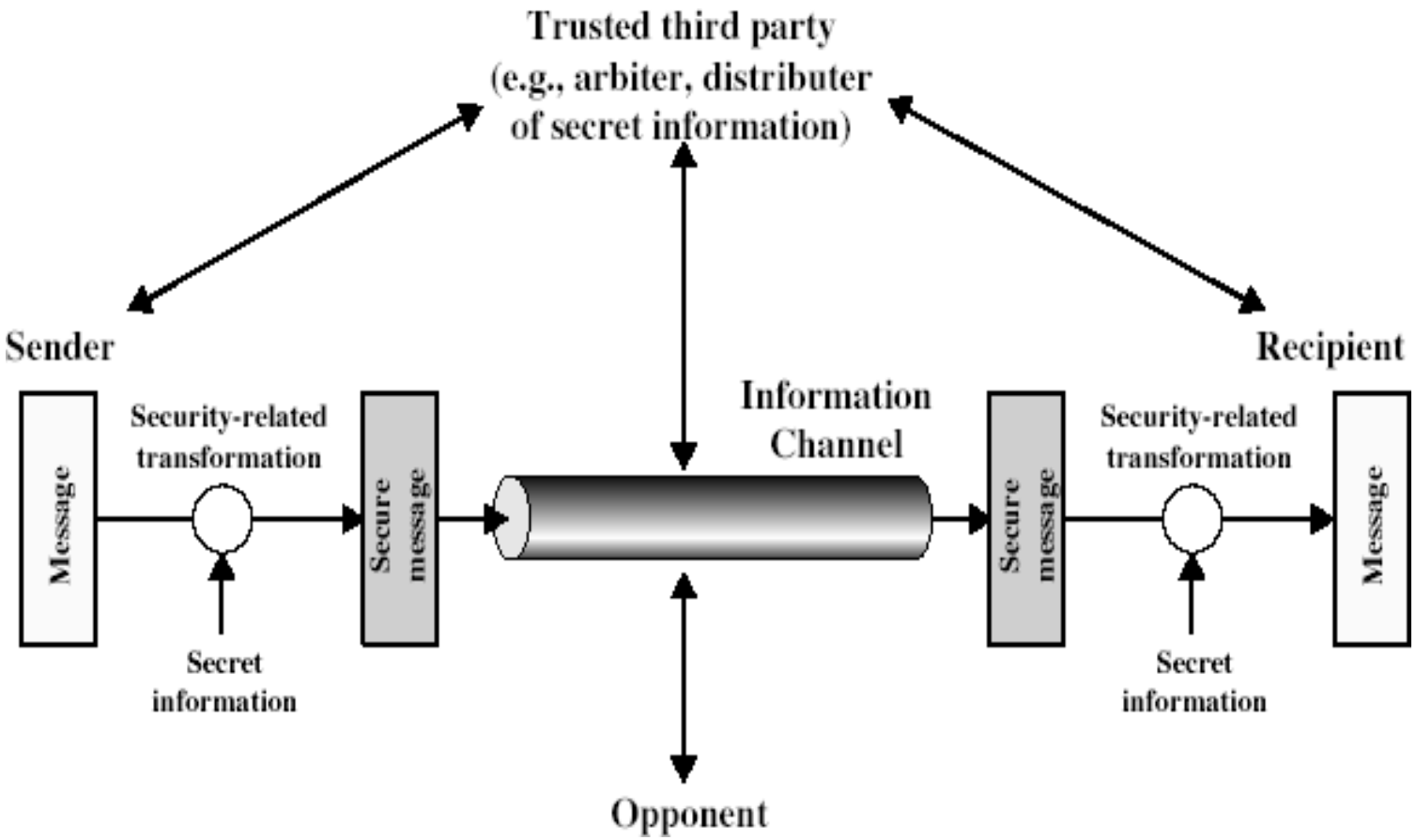
# Security Mechanism

- Feature designed to detect, prevent, or recover from a security attack
- No single mechanism that will support all services required
- However one particular element underlies many of the security mechanisms in use:
- Cryptographic techniques
- Hence our focus on this topic

# Security Mechanisms (X.800)

- specific security mechanisms:
- encipherment, digital signatures, access controls, data integrity, authentication exchange, traffic padding, routing control, notarization
- pervasive security mechanisms:
- trusted functionality, security labels, event detection,
- security audit trails, security recovery

# Model for Network Security



# Model for Network Security

- Using this model requires us to:
  1. Design a suitable algorithm for the security transformation
  2. Generate the secret information (keys) used by the algorithm
  3. Develop methods to distribute and share the secret information
  4. Specify a protocol enabling the principals to use the transformation and secret information for a security service

# Symmetric Encryption

- 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 ender/receiver
- **Encipher (encrypt)** - converting plaintext to ciphertext
- **Decipher (decrypt)** - recovering ciphertext from plaintext
- **Cryptography** - study of encryption rinciples/methods
- **Cryptanalysis (codebreaking)** - study of principles/ methods of deciphering ciphertext *without* knowing key
- **Cryptology** - field of both cryptography and
  - cryptanalysis

# Cryptanalysis

- Objective to recover key not just message
- General approaches:
  - Cryptanalytic attack
  - Brute-force attack

# Cryptanalytic Attacks

- Ciphertext only
- Only know algorithm & ciphertext, is statistical, know or can identify plaintext
- 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

# Classical Substitution Ciphers

- Where letters of plaintext are replaced by other letters or by numbers or symbols
- 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:
  - $c = E(p) = (p + k) \bmod (26)$
  - $p = D(c) = (c - k) \bmod (26)$

# 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

# Playfair Cipher

- Not even the large number of keys in a monoalphabetic cipher provides security
- One approach to improving security was to encrypt multiple letters
- The **Playfair Cipher** is an example
- Invented by Charles Wheatstone in 1854, but named after his friend Baron Playfair

# Play fair Key Matrix

- A 5X5 matrix of letters based on a keyword
- Fill in letters of keyword (sans duplicates)
- Fill rest of matrix with other letters
- Eg. using the keyword MONARCHY

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

# Encrypting and Decrypting

- Plaintext is encrypted two letters at a time
  - if both letters fall in the same row, replace each with letter to right(wrapping back to start from end)
  - if both letters fall in the same column, replace each with the letter below it (again wrapping to top from bottom)
  - otherwise each letter is replaced by the letter in the same row and in the column of the other letter of the pair
  - if a pair is a repeated letter, insert filler like 'X'

# Security of Playfair Cipher

- Security much improved over monoalphabetic
  since have 26 x 26 = 676 diagrams
- Would need a 676 entry frequency table to analyse (verses 26 for a monoalphabetic) and correspondingly more ciphertext was widely used for many years
  - eg. by US & British military in WW1
- it **can** be broken, given a few hundred letters
  since still has much of plaintext structure

# 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

# 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
- MEMATRHTGPRYETEFET
  AAT

# 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

# Steganography

- An alternative to encryption
- Hides existence of message
  - using only a subset of letters/words in a longer message  marked in some way
  - using invisible ink
  - hiding in LSB in graphic image or sound file
- Has drawbacks
  - High overhead to hide relatively few info bits


# UNIT– II
# SYMMETRIC KEY CIPHERS

| CLOs | Course Learning Outcome |
|---|---|
| CLO1 | Understand the block cipher modes of operation for encryption and decryption. |
| CLO2 | Describe the need of stream ciphers in message encryption. |
| CLO3 | Understand the role of elliptic curve cryptography in security. |
| CLO4 | Analyze the drawbacks of RSA and able to design a security algorithm which overcomes that drawbacks. |

# Modern Block Ciphers

- Now look at modern block ciphers
- One of the most widely used types of cryptographic algorithms
- Provide secrecy /authentication services
- Focus on DES (Data Encryption Standard)
- To illustrate block cipher design principles

# Block vs Stream Ciphers

- Block ciphers process messages in 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
- Broader range of applications

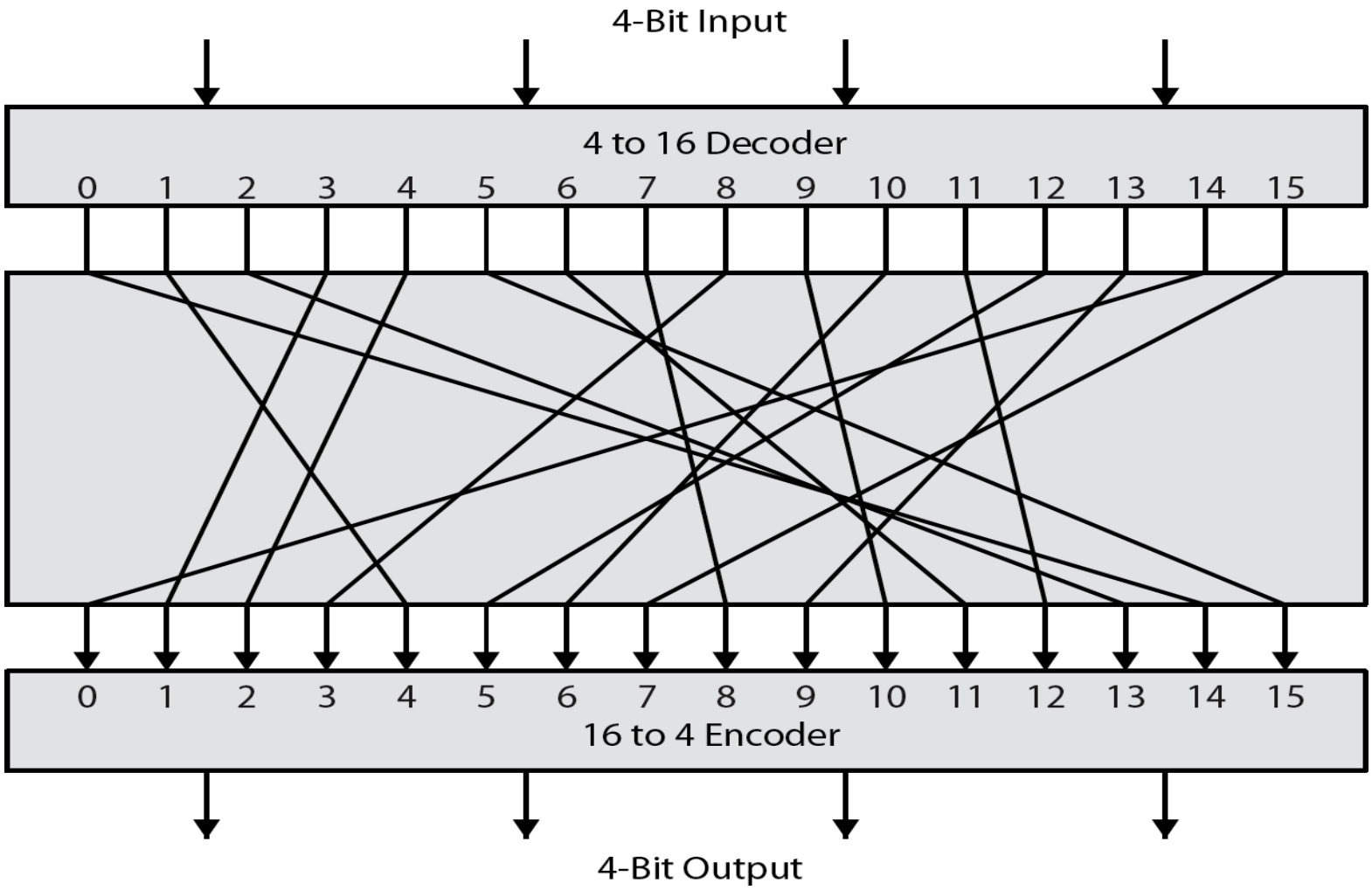
# Block Cipher Principles

- 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 264 entries for a 64-bit block

  Instead create from smaller building blocks

  using idea of a product cipher

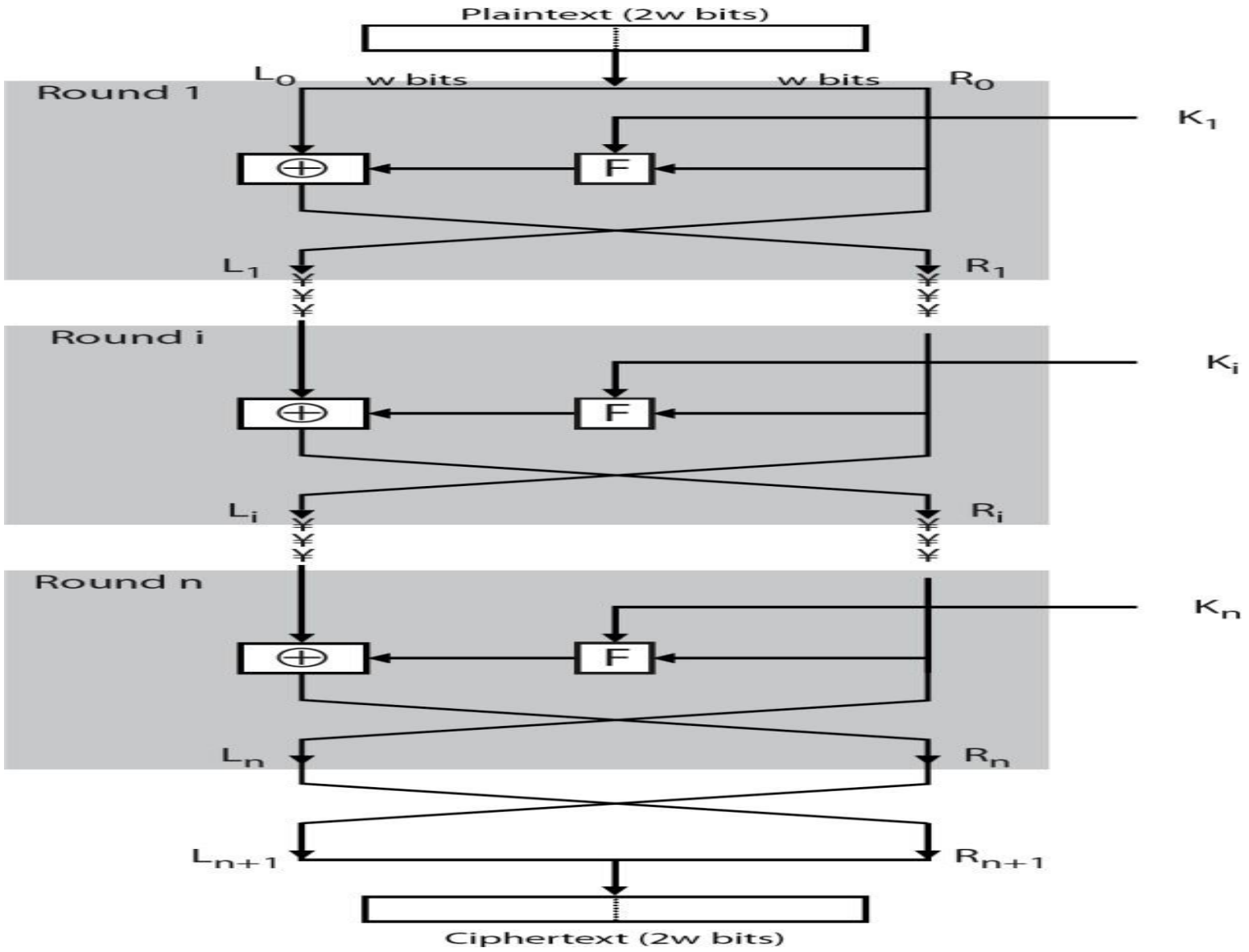
# Ideal Block Cipher



# 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 S-P net concept

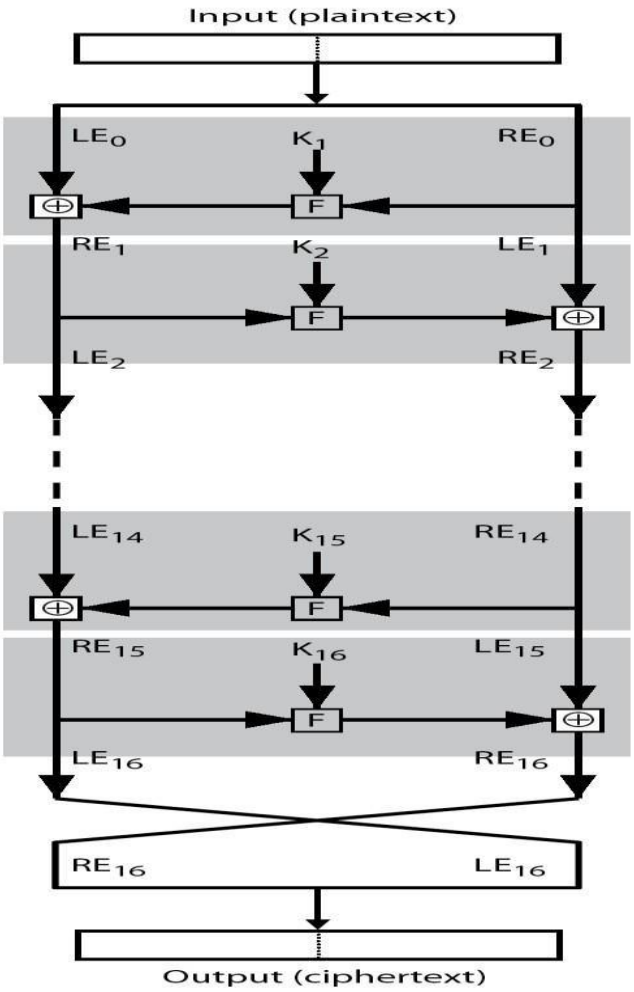
# Feistel Cipher Structure

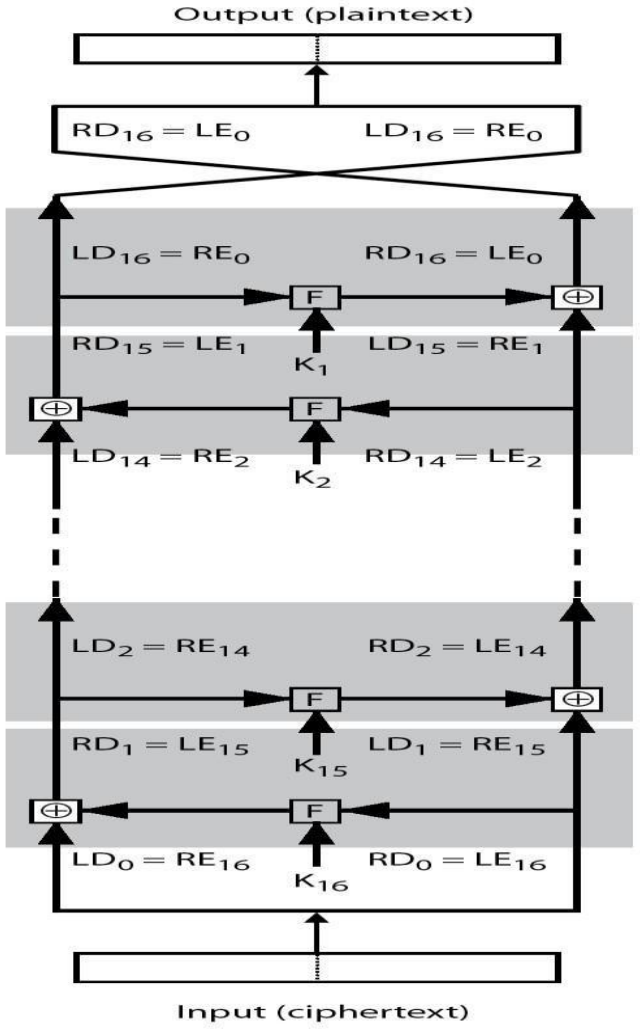


# Feistel Cipher Design Elements

- Block size
- Key size
- Number of rounds
- Subkey generation algorithm
- Rround function
- Fast software en/decryption
- Ease of analysis

# Feistel Cipher Decryption

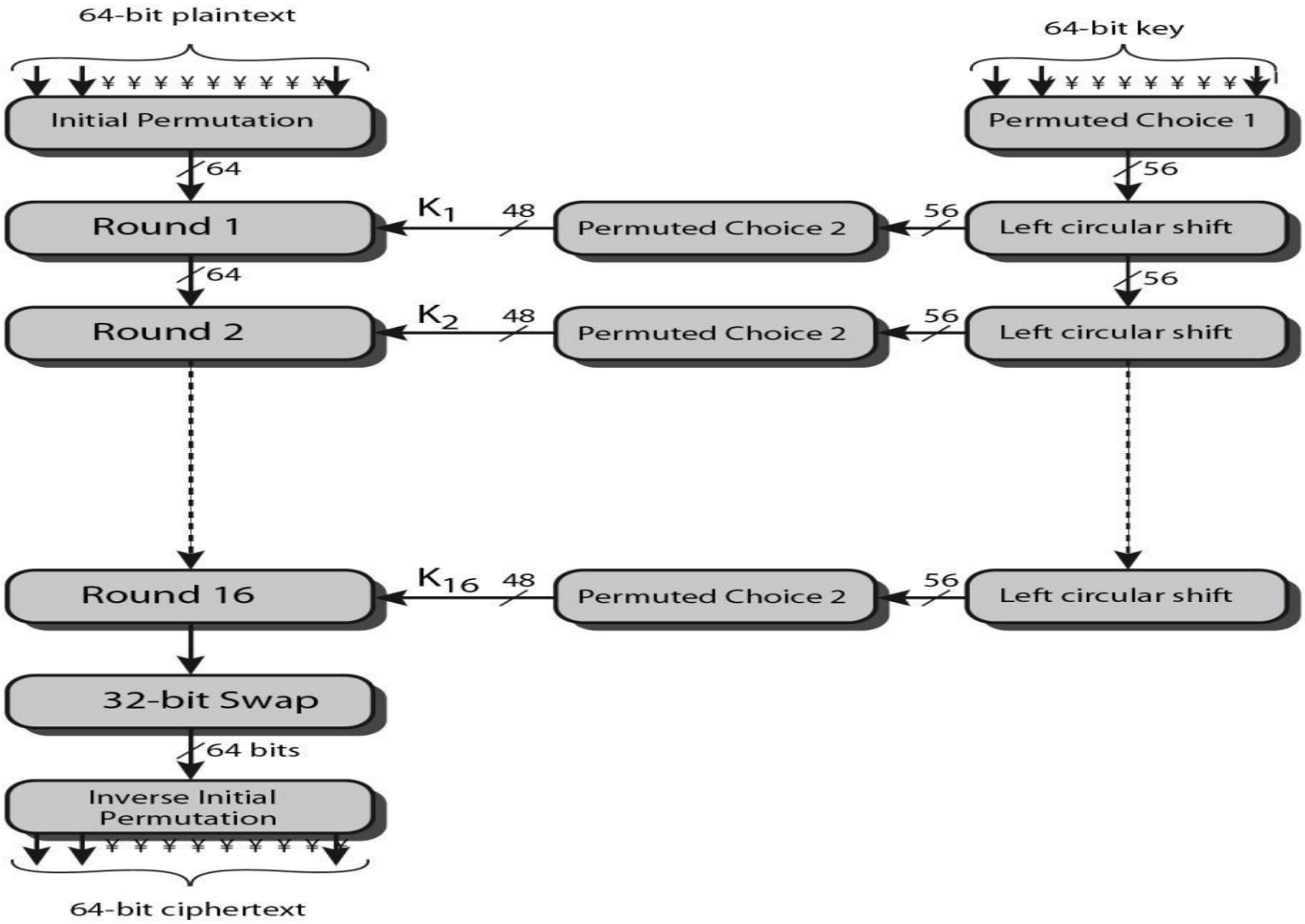


# Data Encryption Standard (DES)

- 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 Encryption Overview



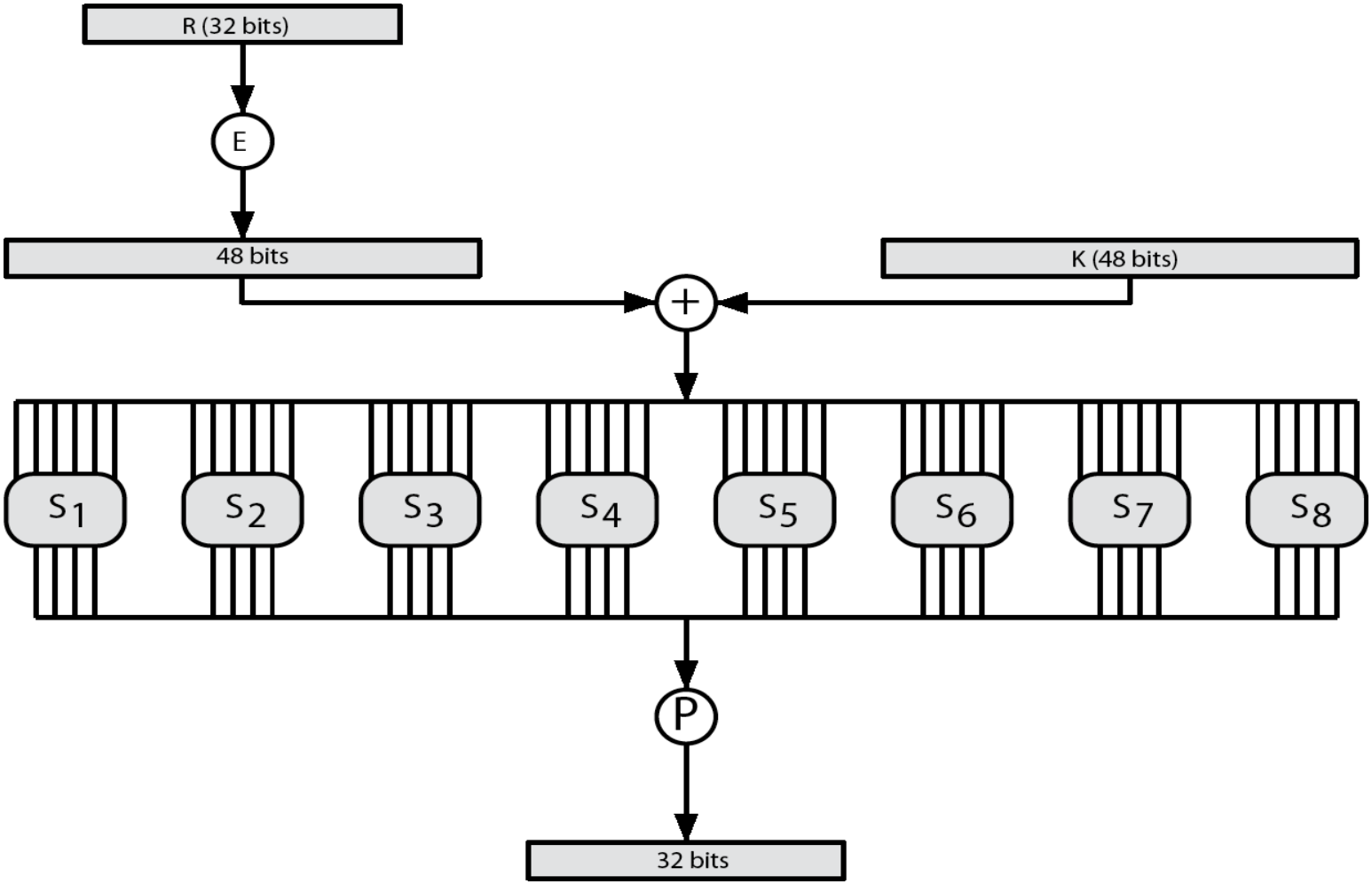
# Initial Permutation IP

- 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)
- Example:
  IP(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)

# DES Round Structure

- Uses two 32-bit L & R halves
- As for any Feistel cipher can describe as:
  - Li = Ri–1
  - Ri = Li–1  F(Ri–1, Ki)
- F takes 32-bit R half and 48-bit subkey:
  - Expands R to 48-bits using perm E
  - Adds to subkey using XOR
  - Passes through 8 S-boxes to get 32-bit result
  - Finally permutes using 32-bit perm P

# DES Round Structure



# Substitution Boxes

- 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 row of 4
  - 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
  - initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves 16 stages consisting of:
    - rotating each half separately either 1 or 2 places depending onthe key rotation schedule K
    - selecting 24-bits from each half & permuting them by PC2 for use in round function F
- Note practical use issues in h/w vs s/w

# DES Decryption

- Decrypt must unwind steps of data computation
- With Feistel design, do encryption steps again using subkeys in reverse order (SK16 … SK1)
  - 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

# 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
- must now consider alternatives to DES

# 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

# Strength of DES – Timing Attacks

- Attacks actual implementation of cipher
- Use knowledge of consequences of implementation to derive information about 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

# 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 in 90’s
- 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 value of the key
- Differential cryptanalysis compares two related pairs of encryptions

# Differential Cryptanalysis Compares Pairs of Encryptions

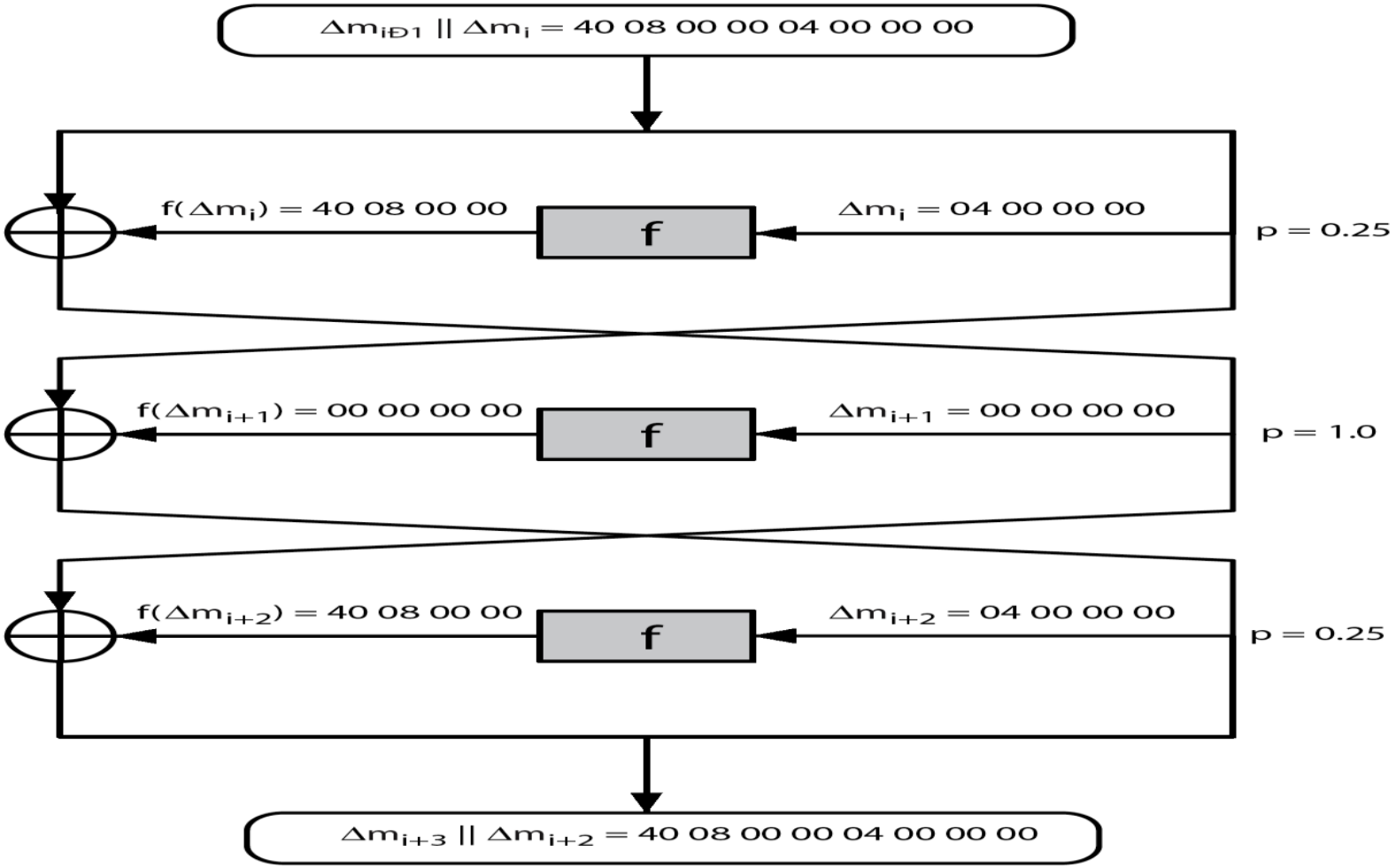
- With a known difference in the input searching for a known difference in output when same subkeys are used

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

# 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 $2^{43}$ known plaintexts, easier but 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: $|p - 1/2|$

# 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)

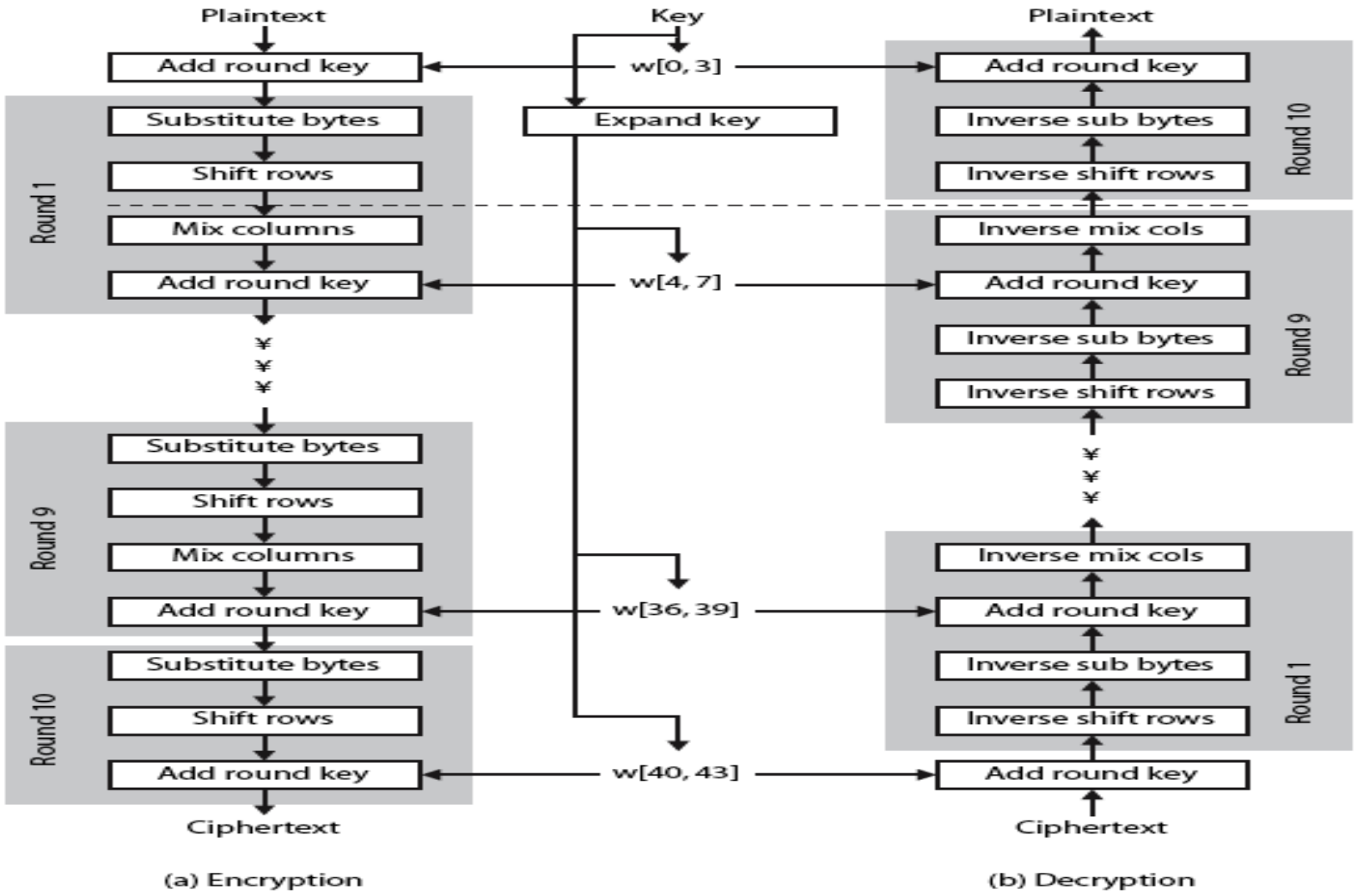
# The AES Cipher - Rijndael

- 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

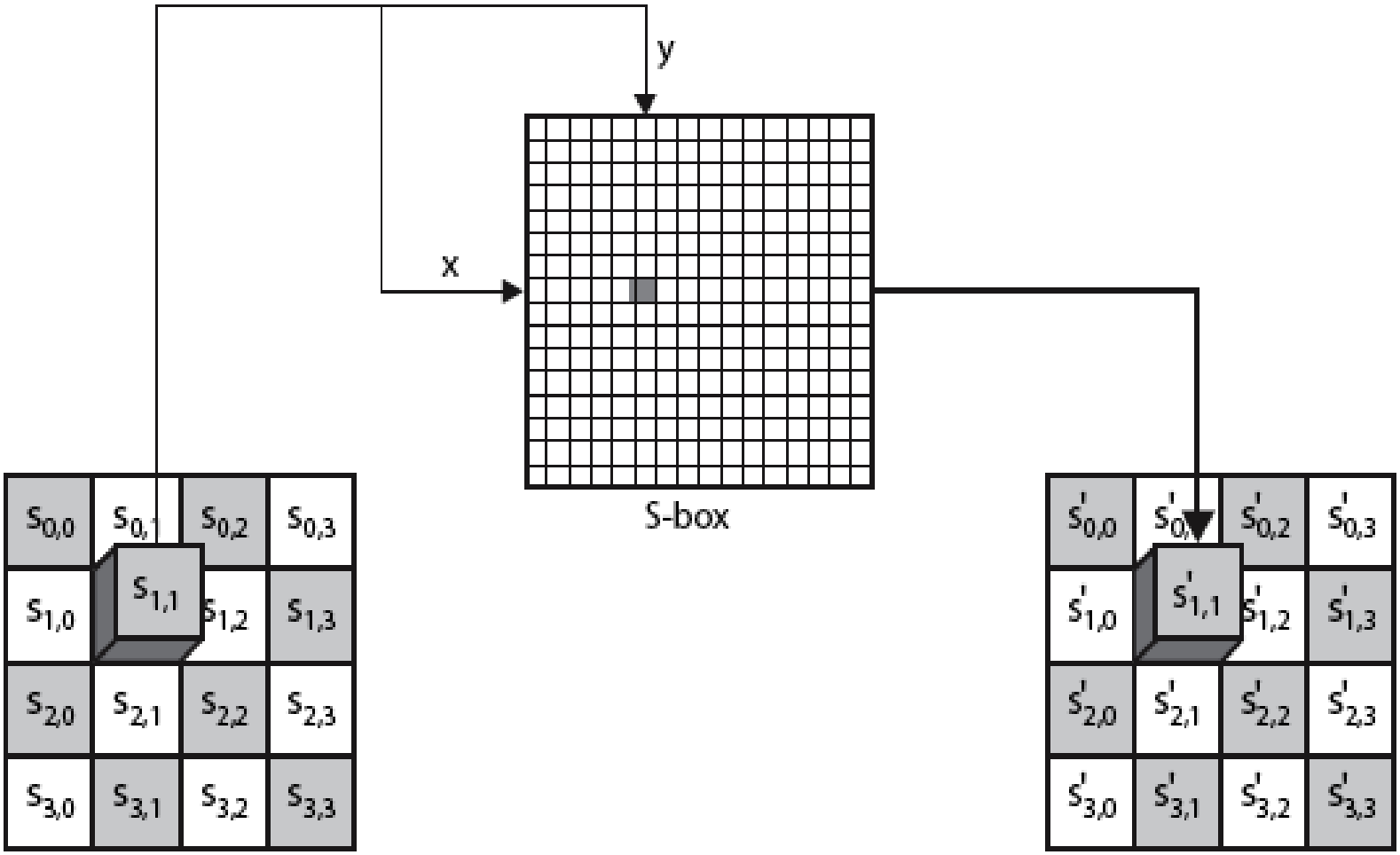
# Rijndael


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

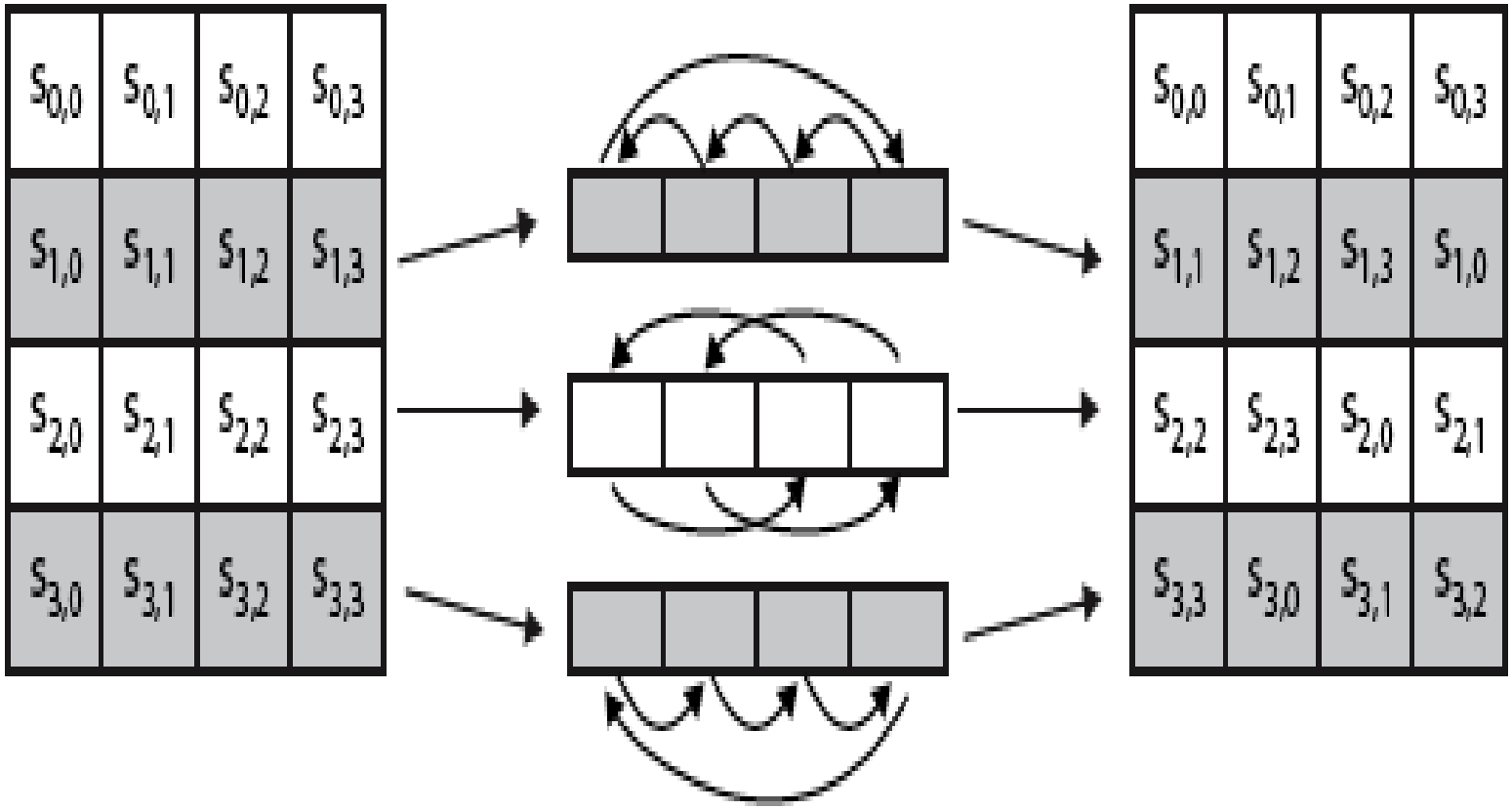
# Byte Substitution



# 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

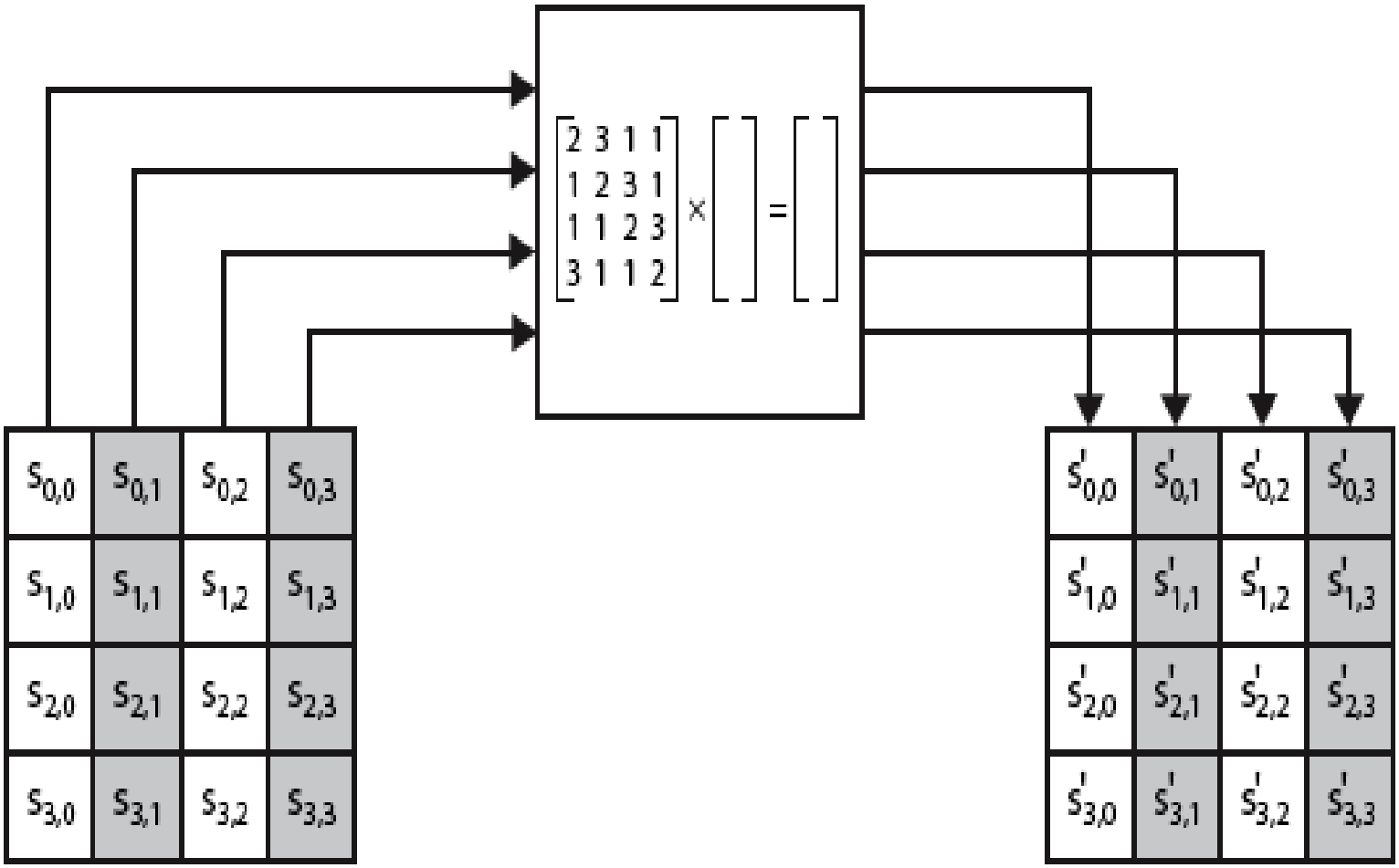

$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,1}$ $S_{1,2}$ $S_{1,3}$ $S_{1,0}$
$S_{2,2}$ $S_{2,3}$ $S_{2,0}$ $S_{2,1}$
$S_{3,3}$ $S_{3,0}$ $S_{3,1}$ $S_{3,2}$

# 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
×
=
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

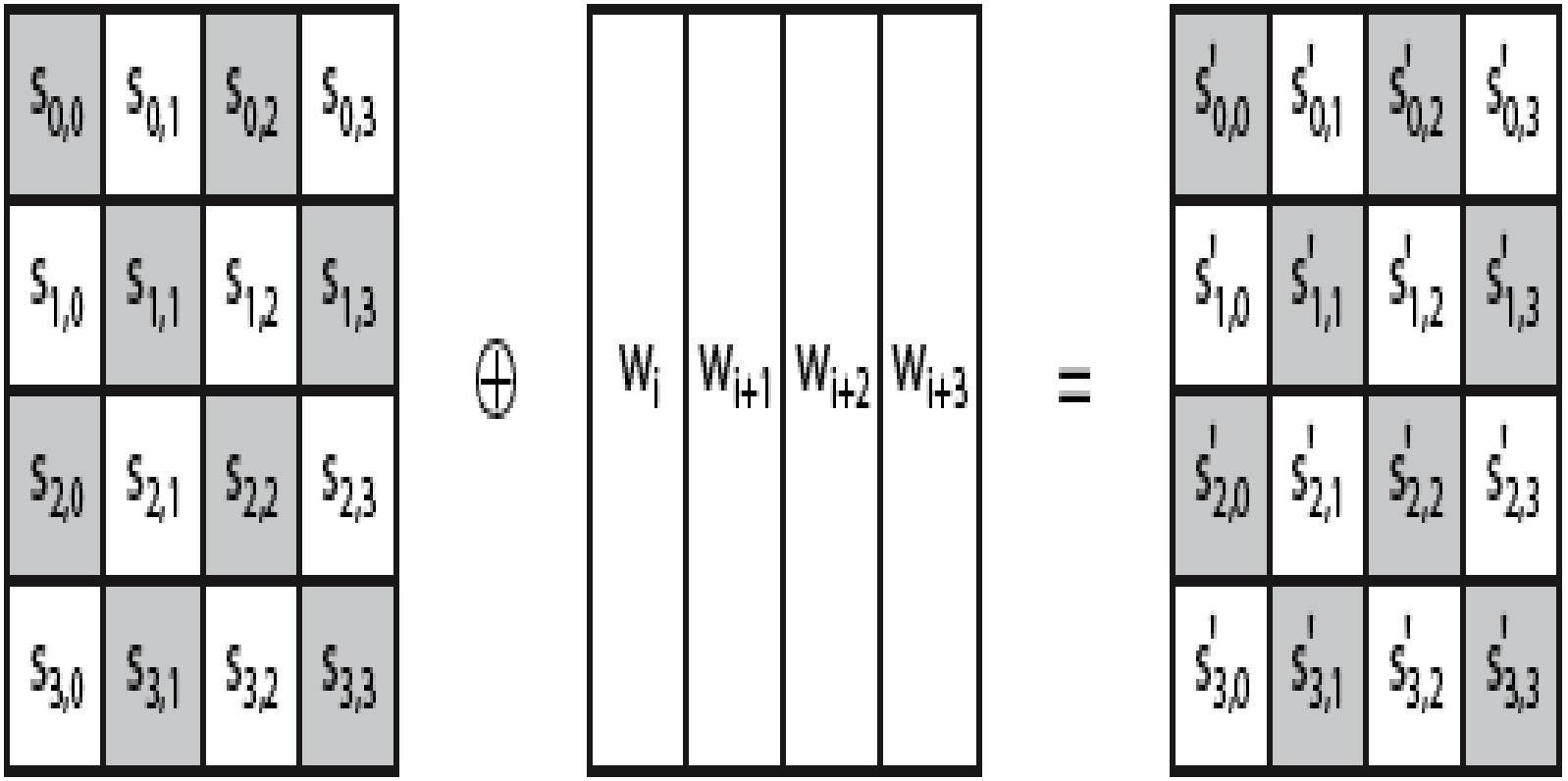
# Mix Columns

- 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(28)
  - and polynomials multiplied modulo (x4+1)

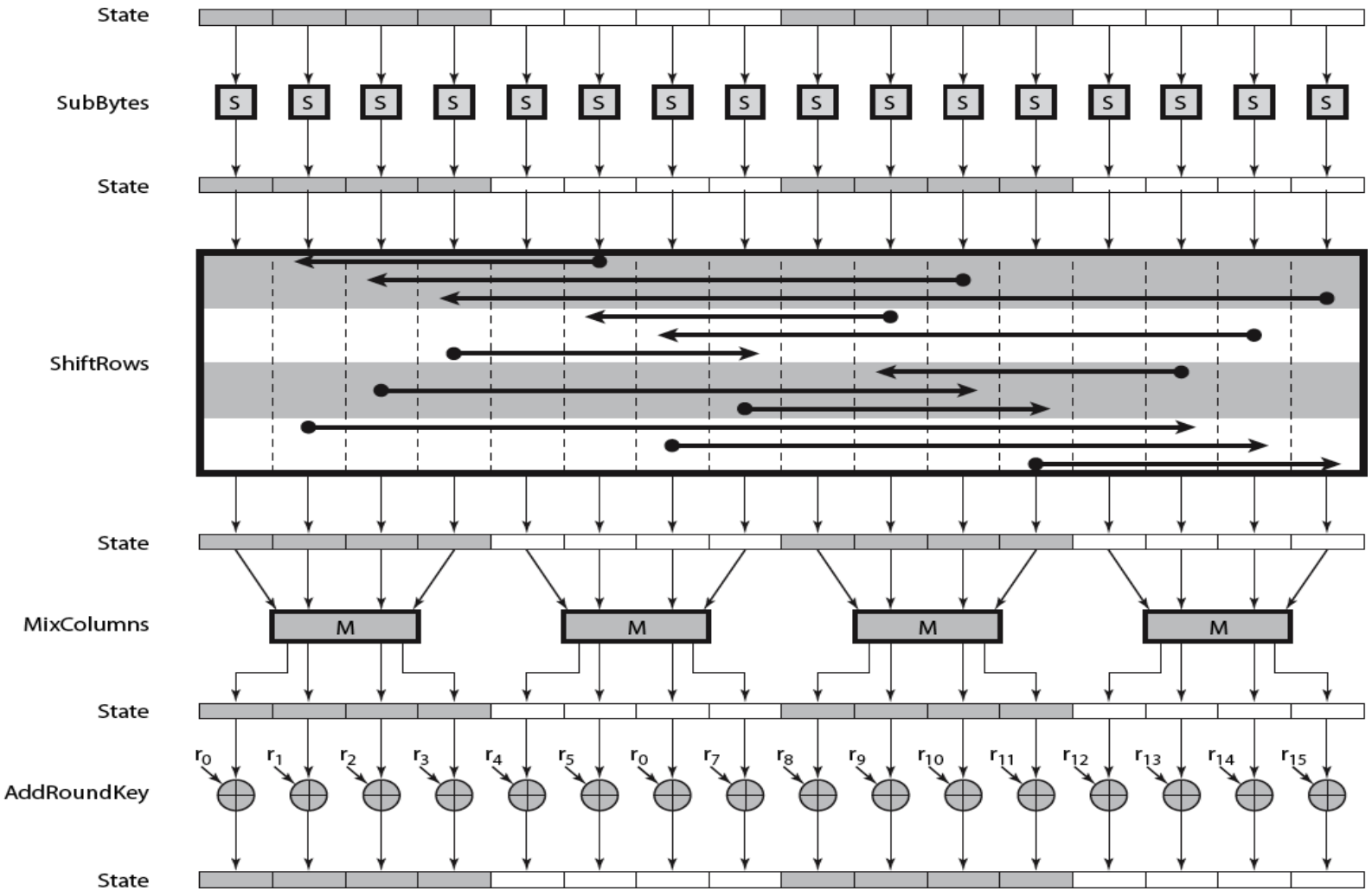
# Add Round Key

- 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



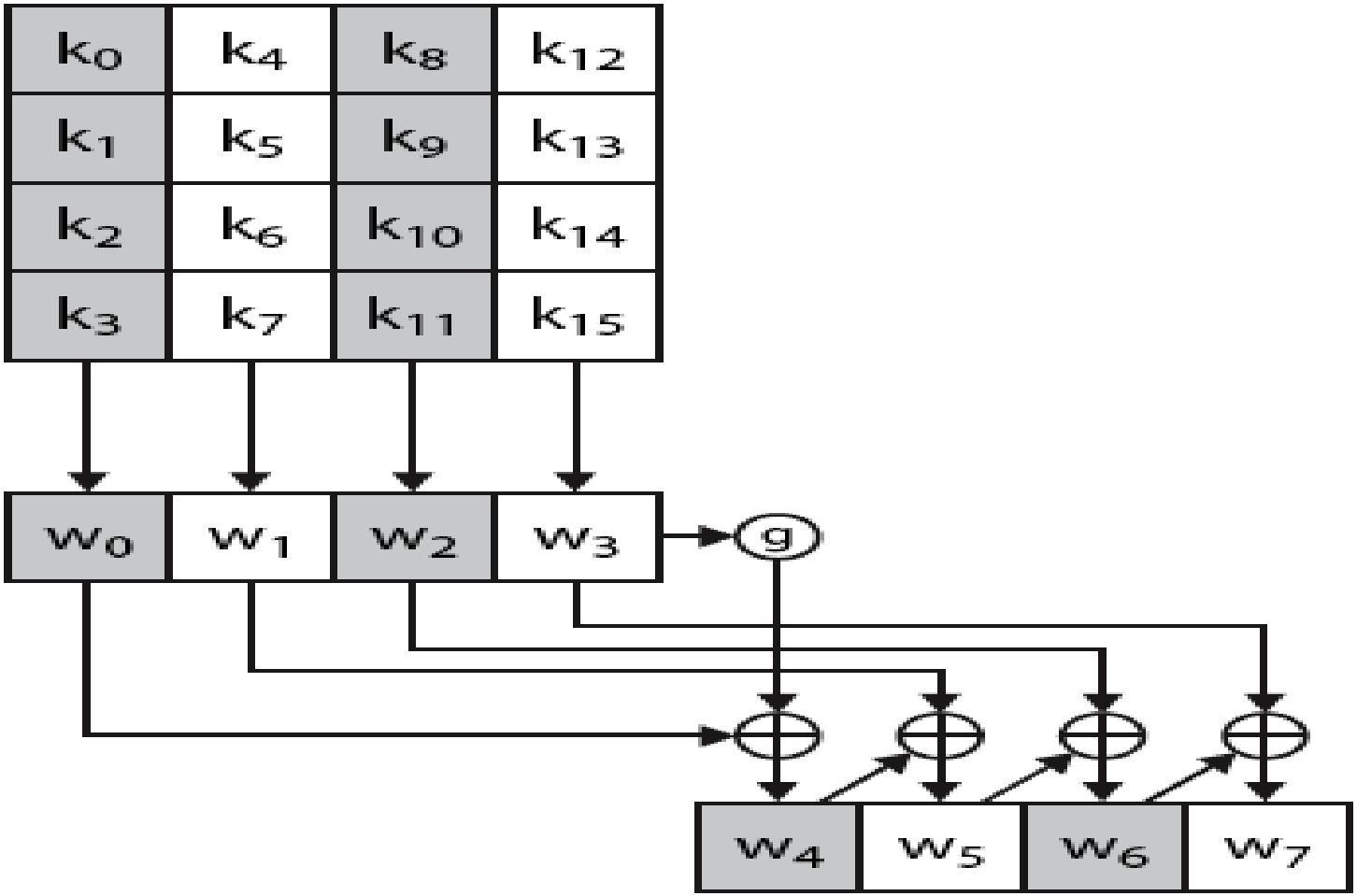
# AES Round


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r0 r1 r2 r3 r4 r5 r0 r7 r8 r9 r10 r11 r12 r13 r14 r15
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



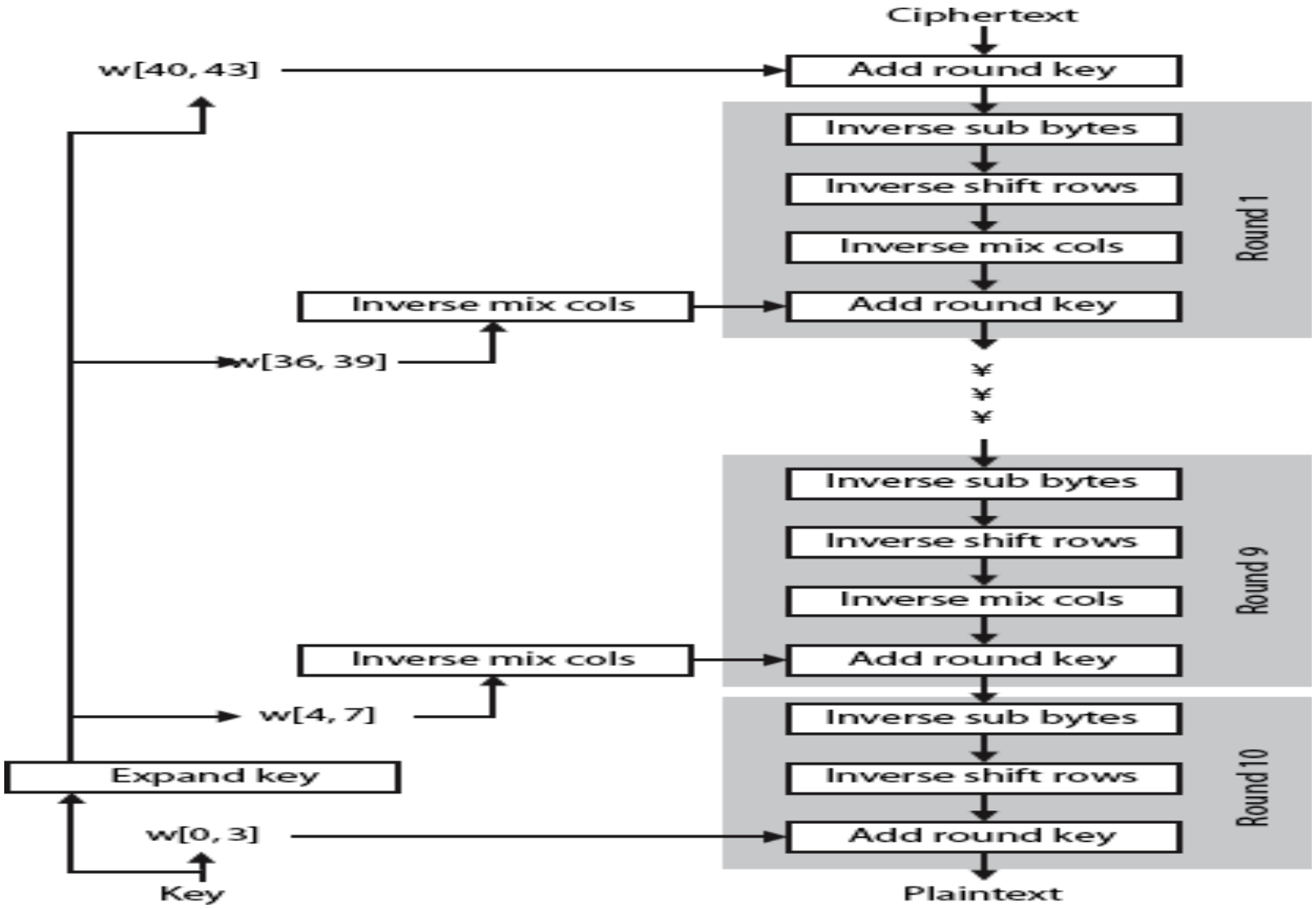
# Key Expansion Rationale

- 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



# Multiple Encryption & DES

- Clear a replacement for DES was needed
  - theoretical attacks that can break it
  - demonstrated exhaustive key search attacks
- AES is a new cipher alternative
- Prior to this alternative was to use multiple encryption with DES implementations
- Triple-DES is the chosen form

# Double-DES?

- Could use 2 DES encrypts on each block
  - C = EK2(EK1(P))
- Issue of reduction to single stage
- and have “meet-in-the-middle” attack
  - works whenever use a cipher twice
  - since X = EK1(P) = DK2(C)
  - attack by encrypting P with all keys and store
  - then decrypt C with keys and match X value
  - can show takes O(256) steps

# Triple-DES with Two-Keys

- Hence must use 3 encryptions
  - would seem to need 3 distinct keys
- But can use 2 keys with E-D-E sequence
  - C = EK1(DK2(EK1(P)))
  - nb 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

# Triple-DES with Three-Keys

- Although are no practical attacks on two-key Triple-DES have some indications
- Can use Triple-DES with Three-Keys to avoid even these
  - C = EK3(DK2(EK1(P)))
- Has been adopted by some Internet applications, eg PGP, S/MIME

# Blowfish

- A symmetric block cipher designed by Bruce Schneier in 1993/94
- Characteristics
  - fast implementation on 32-bit CPUs, 18 clock cycles per byte
  - compact in use of memory, less than 5KB
  - simple structure for analysis/implementation
  - variable security by varying key size Allows tuning for speed/security tradeoff

# Blowfish Key Schedule

- Uses a 32 to 448 bit key
- Used to generate
  - 18 32-bit subkeys stored in P-array: P1 to P18
  - S-boxes stored in Si,j,
    - i=1..4
    - j=0..255

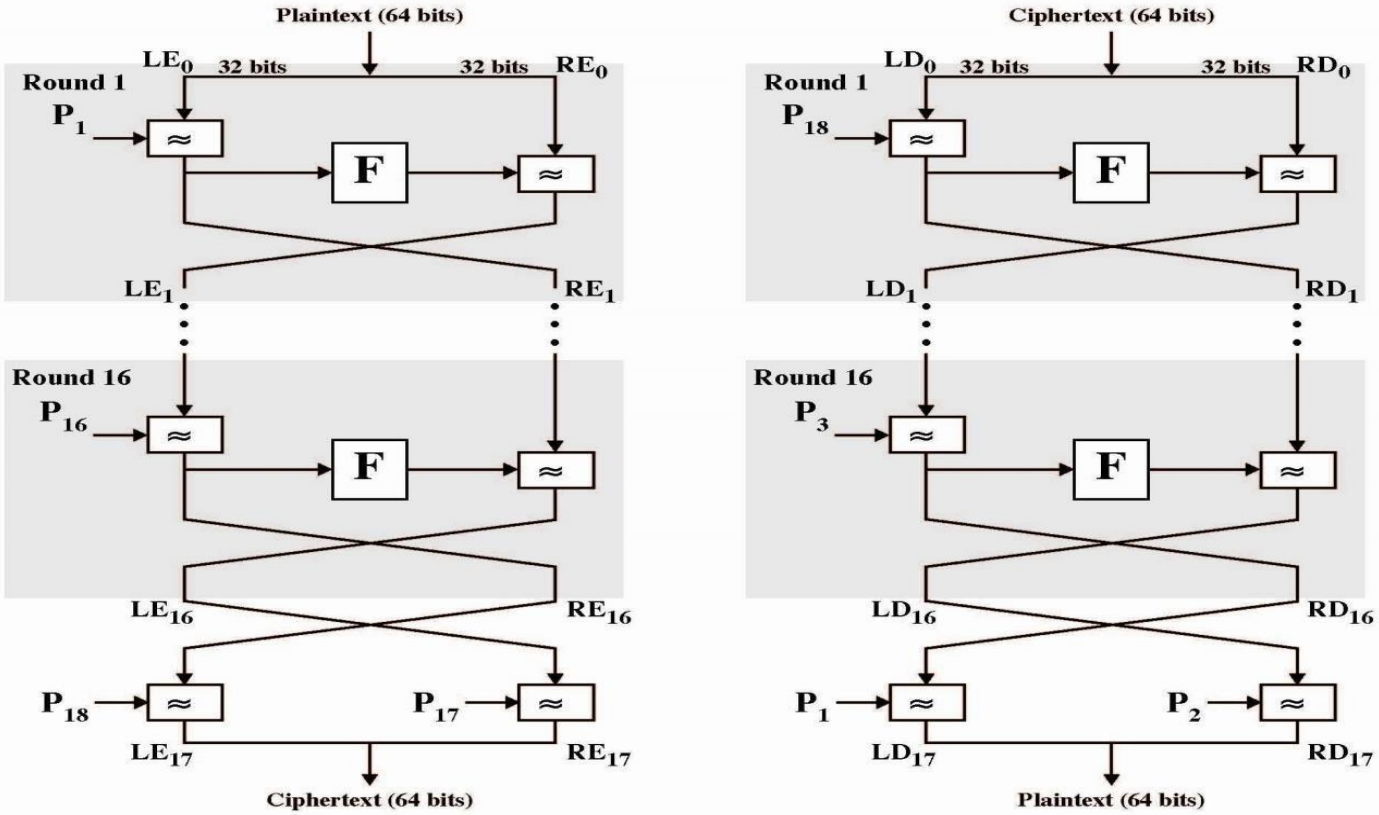



**Figure 6.3 Blowfish Encryption and Decryption**

# Blowfish Encryption

- Uses two primitives: addition & XOR
- Data is divided into two 32-bit halves $L_0$ & $R_0$

- for i = 1 to 16 do

$R_i = L_{i-1}$ XOR $P_i$;

Li = F[$R_i$] XOR $R_{i-1}$;

L17 = $R_{16}$ XOR $P_{18}$;

R17 = $L_{16}$ XOR $i_{17}$;

- where

F[a,b,c,d] = (($S_{1,a}$ + $S_{2,b}$) XOR $S_{3,c}$) + $S_{4,a}$

Break 32-bit $R_i$ into (a,b,c,d)

# Modes of Operation

- Block ciphers encrypt fixed size blocks
  - eg. DES encrypts 64-bit blocks with 56-bit key
- Need some way to en/decrypt arbitrary amounts of data in practise
- ANSI X3.106-1983 Modes of Use (now FIPS 81) defines 4 possible modes
- Subsequently 5 defined for AES & DES

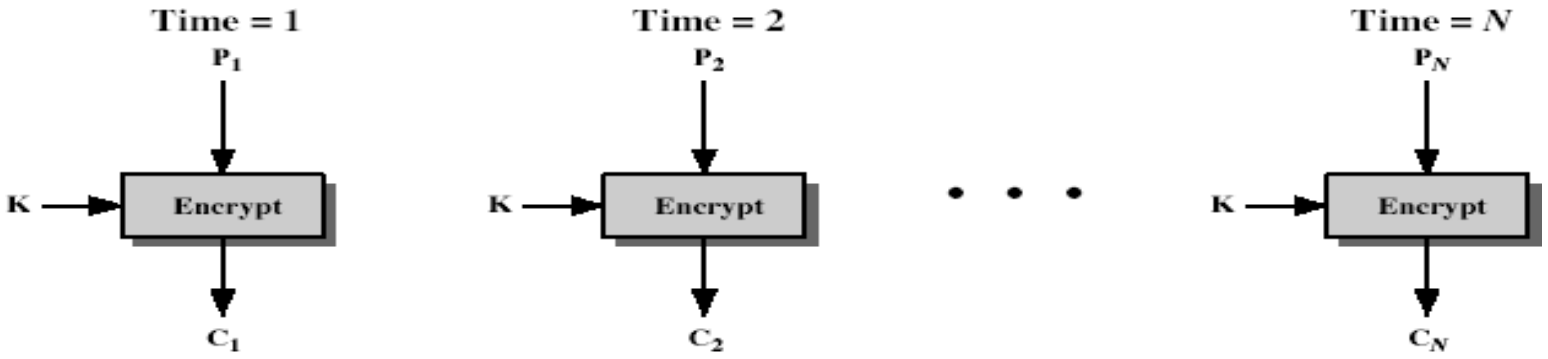
  have block and stream modes

# Electronic Codebook Book (ECB)

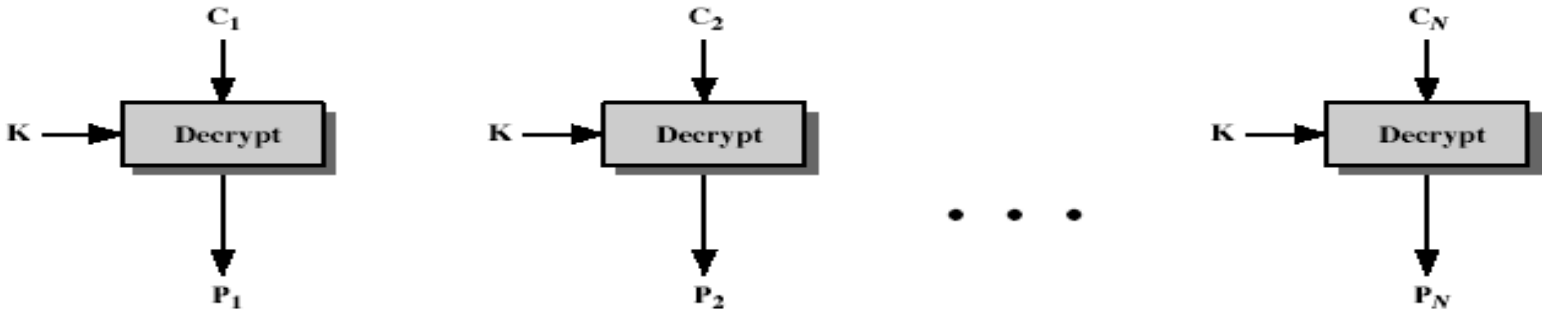
- Message is broken into independent blocks which are encrypted
- Each block is a value which is substituted, like a codebook, hence name
- Each block is encoded independently of the other blocks

  Ci = DESK1(Pi)
- Uses: secure transmission of single values

# Electronic Codebook Book (ECB)




(a) Encryption




(b) Decryption

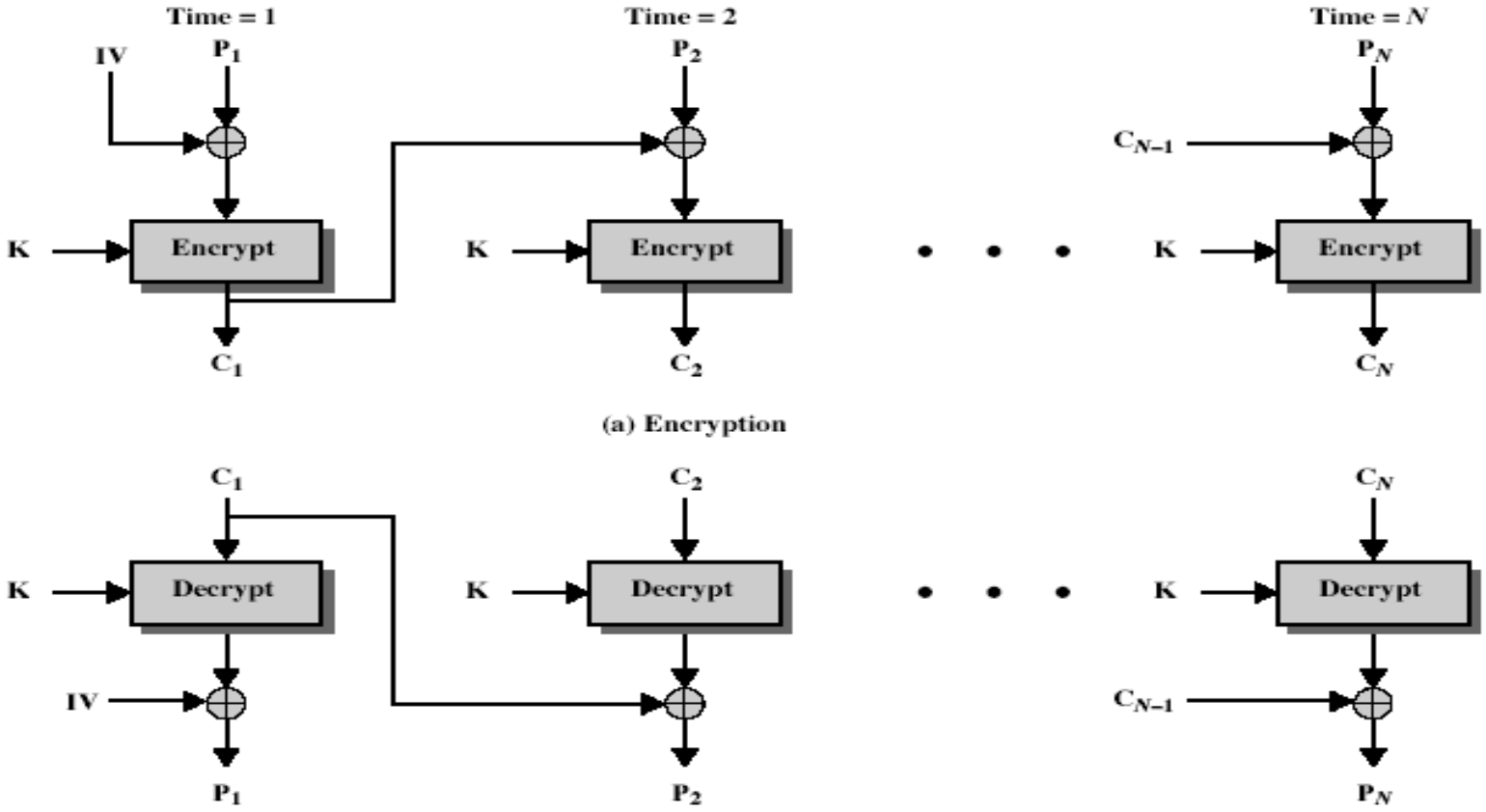
# Advantages and Limitations of ECB

- Message repetitions 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 is due to the encrypted message blocks being independent
- Main use is sending a few blocks of data

# Cipher Block Chaining (CBC)

- Message is broken into blocks
- Linked together in encryption operation
- Each previous cipher blocks is chained with current plaintext block, hence name
- Use Initial Vector (IV) to start process
  - Ci = DESK1(Pi XOR Ci-1)
  - C-1= IV
- Uses: bulk data encryption, authentication

# Cipher Block Chaining (CBC)



# Message Padding

- At end of message must handle a possible last short block
  - which is not as large as blocksize of cipher
  - pad either with known non-data value (eg nulls)
  - or pad last block along with count of pad size
    - eg. [ b1 b2 b3 0 0 0 0 5]
    - means have 3 data bytes, then 5 bytes pad+count
  - this may require an extra entire block over those in message
- There are other, more esoteric modes, which avoid the need for an extra block

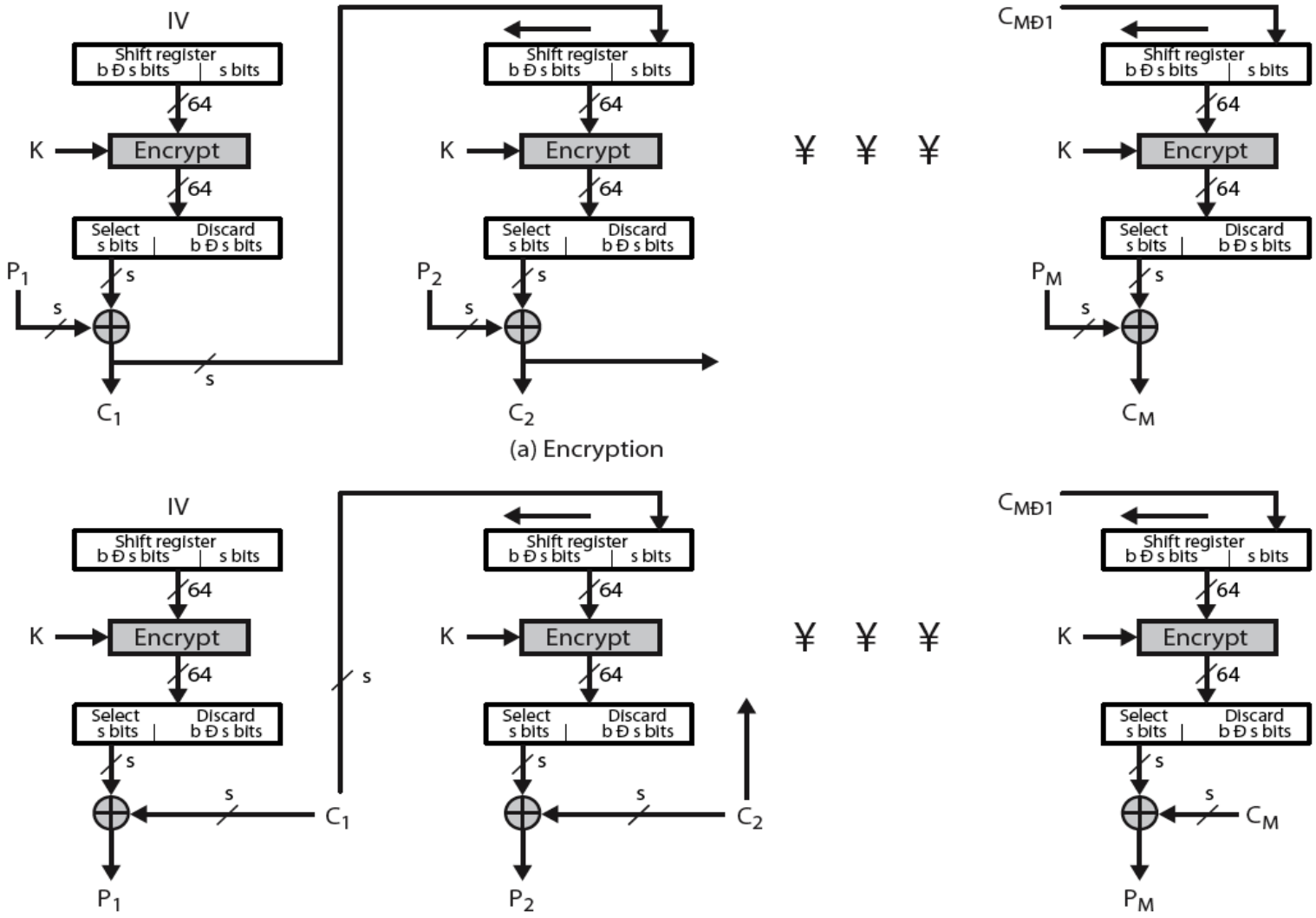
# Advantages and Limitations of CBC

- Aciphertext block depends on **all** blocks before it
- Any change to a block affects all following ciphertext blocks
- Need Initialization Vector (IV)
  - which must be known to sender & receiver
  - if sent in clear, attacker can change bits of first block, and change IV to compensate hence IV must either be a fixed value (as in EFTPOS)
  - or must be sent encrypted in ECB mode before rest of message

# 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, 64 or 128 etc) to be feed back
  - denoted CFB-1, CFB-8, CFB-64, CFB-128 etc
- Most efficient to use all bits in block (64 or 128)
  - Ci = Pi XOR DESK1(Ci-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
  - Ci= Pi  XOR Oi
  - Oi= DESK1(Oi-1)  O-1
    = IV
- Uses: stream encryption on noisy channels

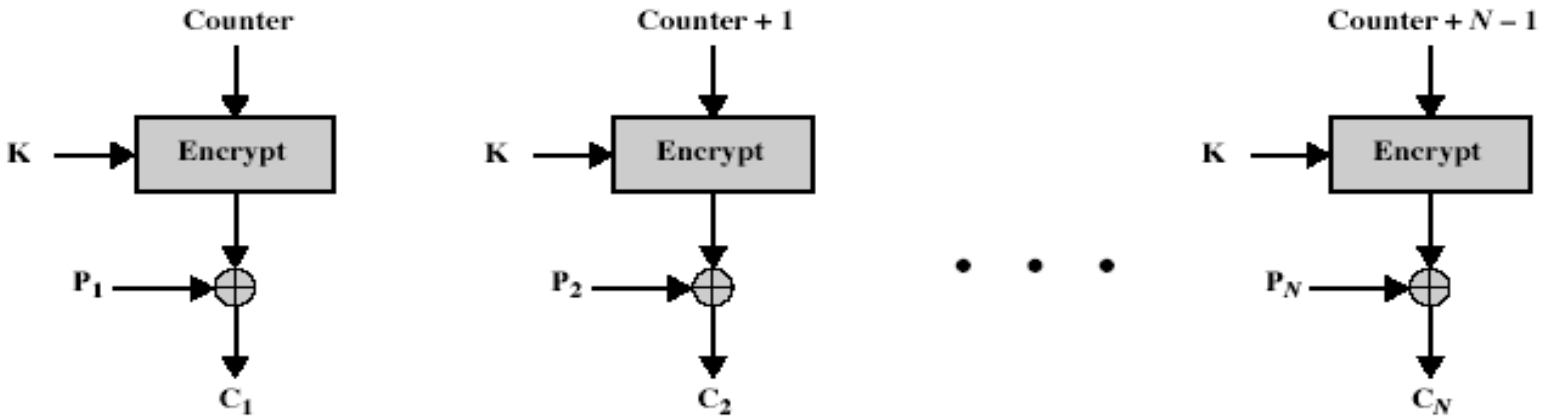
# Advantages and Limitations of OFB

- Bit errors do not propagate
- More vulnerable to message stream modification
- Avariation of a Vernam cipher
  - hence must never reuse the same sequence (key+IV)
- Sender & receiver must remain in sync
- Originally specified with m-bit feedback
- Subsequent research has shown that only full block feedback (ie CFB-64 or CFB-128) should ever be used

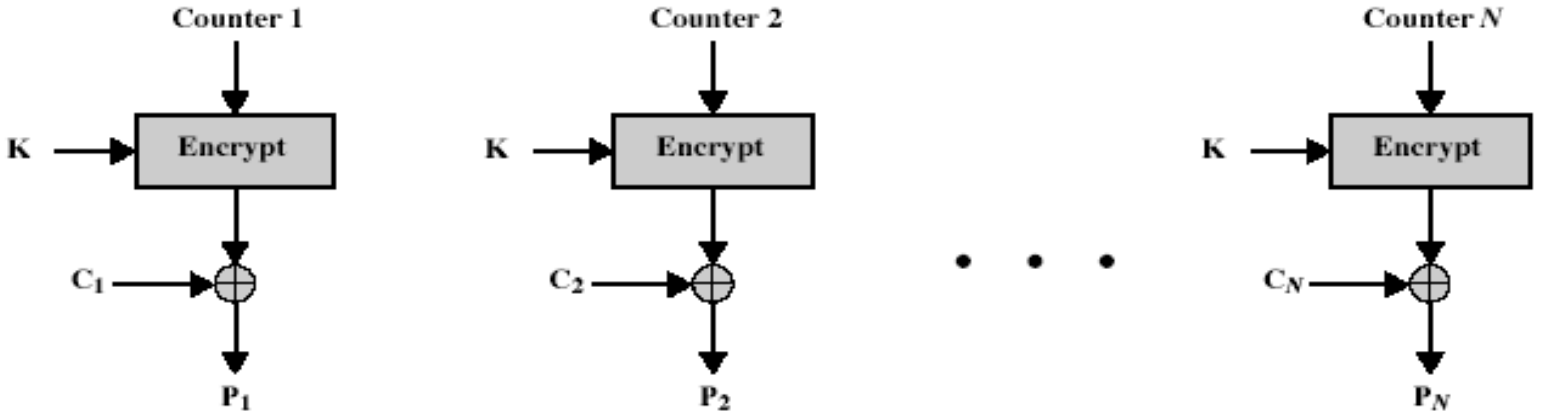
# Counter (CTR)

- 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)
  - Ci = Pi XOR Oi
  - Oi= DESK1(i)
- Uses: high-speed network encryptions

# Counter (CTR)




(a) Encryption




(b) Decryption

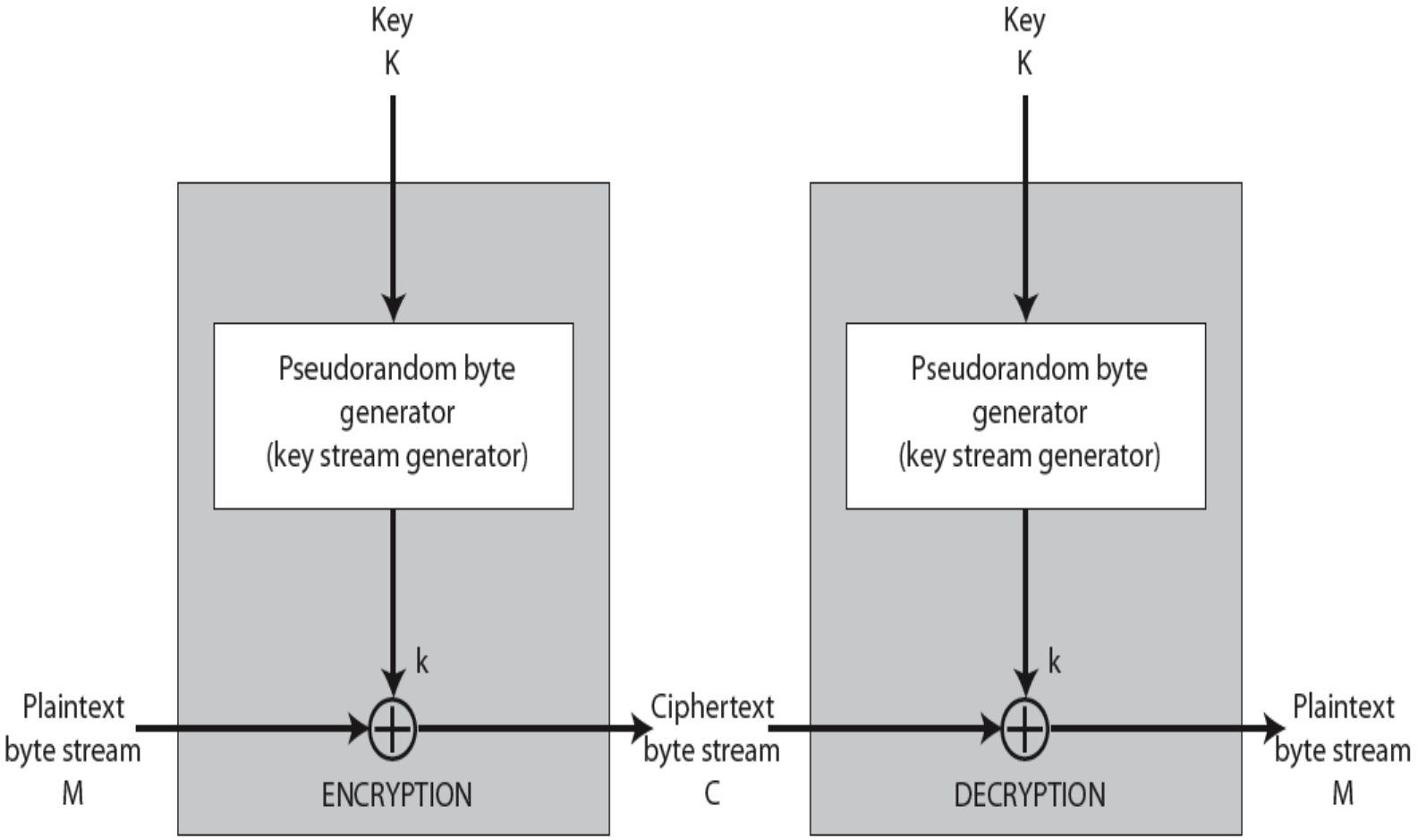
# Advantages and Limitations of CTR

- Efficiency
  - can do parallel encryptions in h/w or s/w
  - can preprocess 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)

# Stream Ciphers

- Process message bit by bit (as a stream)
- Have a pseudo random keystream
- Combined (XOR) with plaintext bit by bit
- Randomness of stream key completely destroys statistically properties in message
  - Ci = Mi XOR StreamKeyi
- But must never reuse stream key
  - otherwise can recover messages (cf book cipher)

# Stream Cipher Structure



# Stream Cipher Properties

- Some design considerations are:
  - long period with no repetitions
  - statistically random
  - depends on large enough key
  - large linear complexity
- Properly designed, can be as secure as a block cipher with same size key but usually simpler & faster

# RC4

- A proprietary cipher owned by RSA DSI
- Another Ron Rivest design, simple but effective
- Variable key size, byte-oriented stream cipher
- Widely used (web SSL/TLS, wireless WEP)
- Key forms random permutation of all 8-bit values
- Uses that permutation to scramble input info  processed a byte at a time

# RC4 Key Schedule

- Starts with an array S of numbers: 0..255
  use key to well and truly shuffle
- S forms **internal state** of the cipher

```
      for i =   0 to 255 do
       S[i] =   i
       T[i] =   K[i mod keylen])
      j = 0

        for i = 0 to 255 do
         j = (j + S[i] + T[i]) (mod          256)

swap (S[i], S[j])
```

# RC4 Encryption

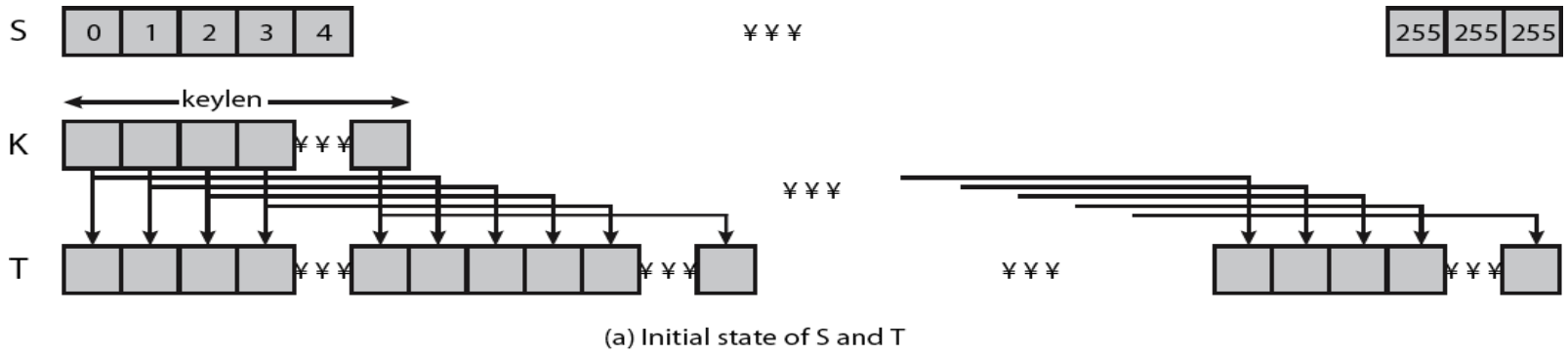
- Encryption continues shuffling array values
- Sum of shuffled pair selects "stream key" value from permutation
- XOR S[t] with next byte of message to en/decrypt
  - i = j = 0
  - for each message

    byte Mi  i = (i + 1)

    (mod 256)
    - j = (j + S[i]) (mod 256)
    - swap(S[i], S[j])
    - t = (S[i] + S[j]) (mod 256)

    Ci= Mi XOR S[t]

# RC4 Overview



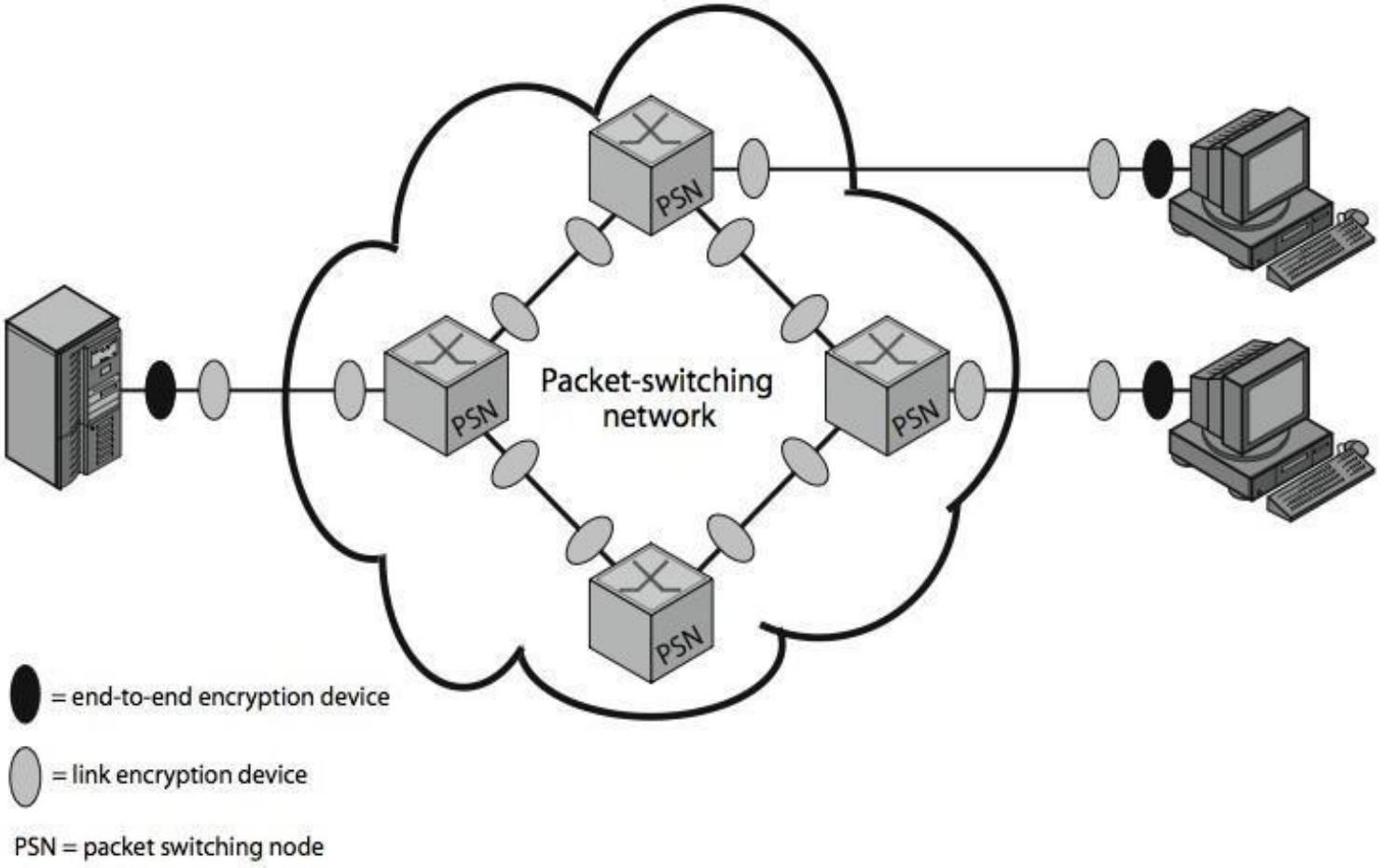
# RC4 Security

- Claimed secure against known attacks
  - have some analyses, none practical
- Result is very non-linear
- Since RC4 is a stream cipher, must never reuse a key
- Have a concern with WEP, but due to key handling rather than RC4 itself

# Placement of Encryption

- Have two major placement alternatives
- **Link encryption**
  - encryption occurs independently on every link
  - implies must decrypt traffic between links
  - requires many devices, but paired keys
- End-to-end encryption
  - encryption occurs between original source and final destination
  - need devices at each end with shared keys

# Placement of Encryption



# Placement of Encryption

- When using end-to-end encryption must leave headers in clear
  - so network can correctly route information
- Hence although contents protected, traffic pattern flows are not
- Ideally want both at once
  - end-to-end protects data contents over entire path and provides authentication
  - link protects traffic flows from monitoring

# Placement of Encryption

- Can place encryption function at various layers in OSI Reference Model
  - link encryption occurs at layers 1 or 2
  - end-to-end can occur at layers 3, 4, 6, 7
  - as move higher less information is encrypted but it is more secure though more complex with more entities and keys

# Private-Key Cryptography

- Traditional **private/secret/single key**
  - cryptography uses **one** key
- Shared by both sender and receiver
- If this key is disclosed communications are compromised
- Also is **symmetric**, parties are equal
- Hence does not protect sender from receiver forging a message & claiming is sent by sender

# Public-Key Cryptography

- Probably most significant advance in the 3000 year history of cryptography
- Uses **two** keys – a public & a private key
- **Asymmetric** since parties are **not** equal
- Uses clever application of number theoretic concepts to function
- Complements **rather than** replaces private key crypto

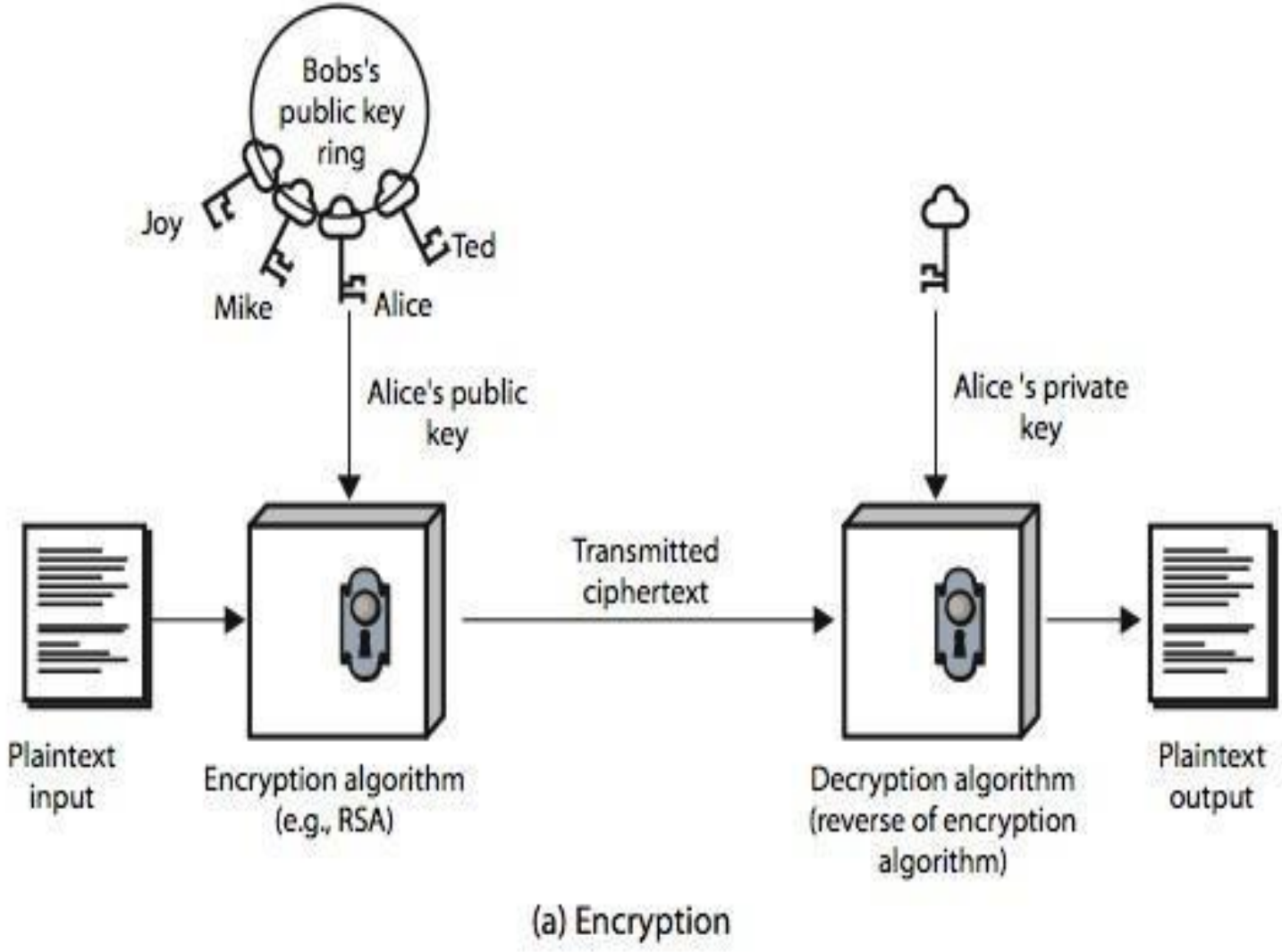
# Why Public-Key Cryptography?

- Developed to address two key issues:
  - key distribution – how to have secure communications in general without having to trust a KDC with your key
  - digital signatures – how to verify a message comes intact from the claimed sender
- Public invention due to Whitfield Diffie & Martin Hellman at Stanford Uni in 1976
  - known earlier in classified community

# Public-Key Cryptography

- Public-key/two-key/asymmetric cryptography involves the use of two keys:
  - a public-key, which may be known by anybody, and can be used to encrypt messages, and verify signatures
  - a private-key, known only to the recipient, used to decrypt messages, and sign (create) signatures
- Is asymmetric because
  - those who encrypt messages or verify signatures cannot decrypt messages or create signatures

# Public-Key Cryptography

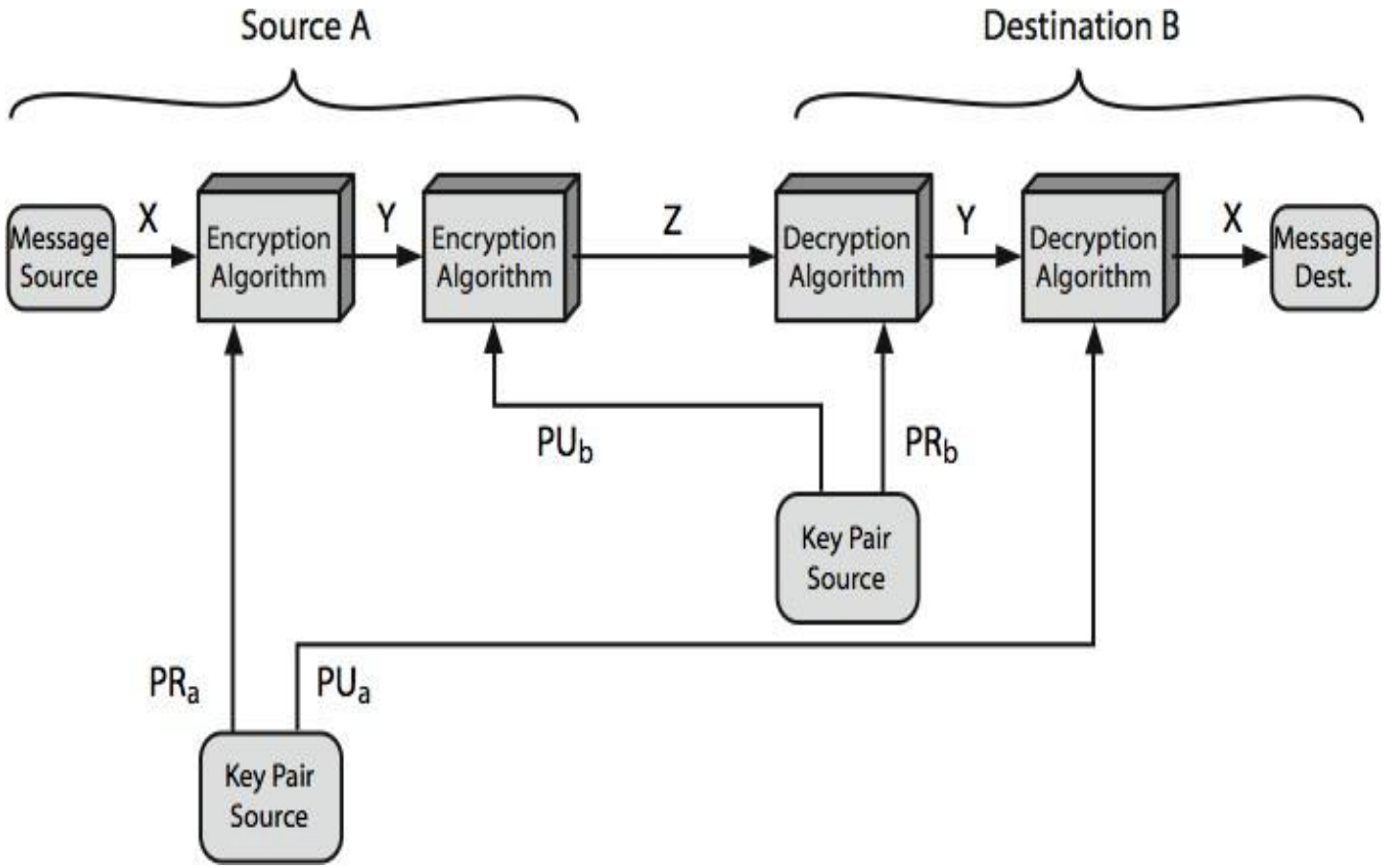



(a) Encryption

# Public-Key Characteristics

- **Public-Key algorithms rely on two keys where:**
  - **it is computationally infeasible to find decryption key knowing only algorithm & encryption key**
  - **it is computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known**
  - **either of the two related keys can be used for encryption, with the other used for decryption (for some algorithms)**

# Public-Key Cryptosystems



# Public-Key Applications

- Can classify uses into 3 categories:
  - encryption/decryption (provide secrecy)
  - digital signatures (provide authentication)
  - key exchange (of session keys)
- Some algorithms are suitable for all uses, others are specific to one

# Security of Public Key Schemes

- Like private key schemes brute force exhaustive search attack is always theoretically possible
- But keys used are too large (>512bits)
- Security relies on a large enough difference in difficulty between easy (en/decrypt) and hard (cryptanalyse) problems
- More generally the hard problem is known, but is made hard enough to be impractical to break
- Requires the use of very large numbers hence is slow compared to private key schemes

# RSA

- By Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key scheme
- Based on exponentiation in a finite (Galois) field over integers modulo a prime
  - nb. exponentiation takes O((log n)3) operations (easy)
- Uses large integers (eg. 1024 bits)
- Security due to cost of factoring large numbers
  - nb. factorization takes O(e log n log log n) operations (hard)

# RSA Key Setup

- Each user generates a public/private key pair by:
- selecting two large primes at random – p,q
- Computing their system modulus n=p.q
  - note ø(n)=(p-1)(q-1)
- Selecting at random the encryption key e
  where 1<e<ø(n), gcd(e,ø(n))=1
- Solve following equation to find decryption key d
  - e.d=1 mod ø(n) and 0≤d≤n
- Publish their public encryption key: PU={e,n}
- Keep secret private decryption key: PR={d,n}

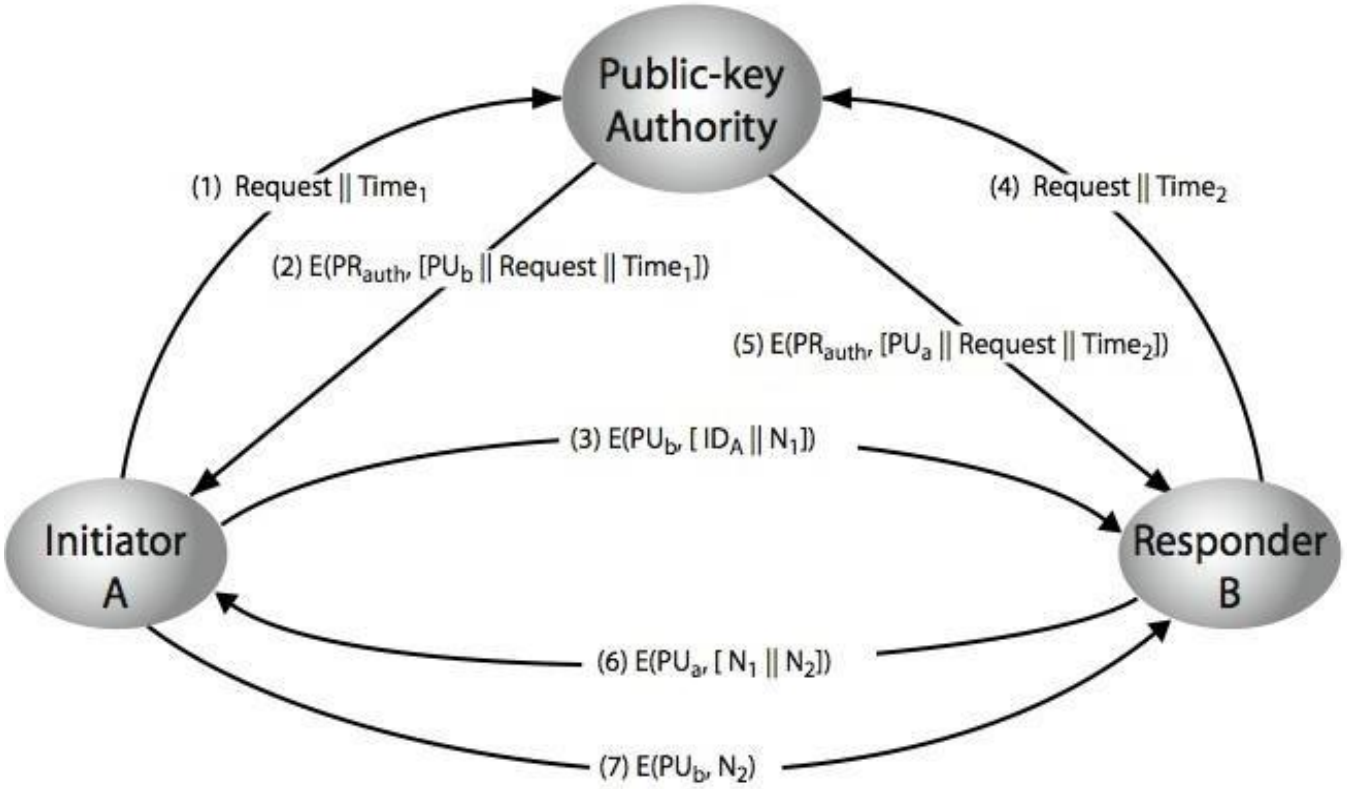
# Key Management

- Public-key encryption helps address key distribution problems
- Have two aspects of this:
  - distribution of public keys
  - use of public-key encryption to distribute secret keys

# Distribution of Public Keys

- Can be considered as using one of:
  - public announcement
  - publicly available directory
  - public-key authority
  - public-key certificates

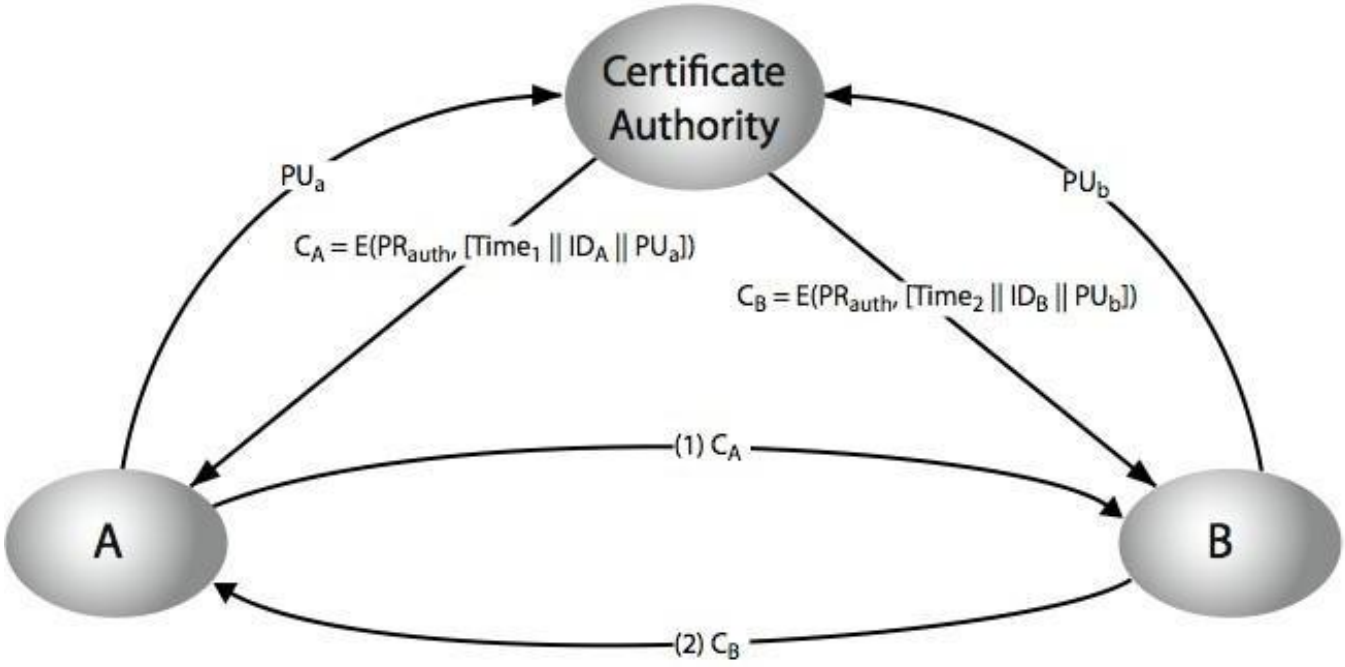
# Public-Key Authority



# Public-Key Certificates

- Certificates allow key exchange without real-time access to public-key authority
- A certificate binds identity to public key
  - usually with other info such as period of validity, rights of use etc
- With all contents signed by a trusted Public-Key or Certificate Authority (CA)
- Can be verified by anyone who knows the public-key authorities public-key

# Public-Key Certificates



# Public-Key Distribution of Secret Keys

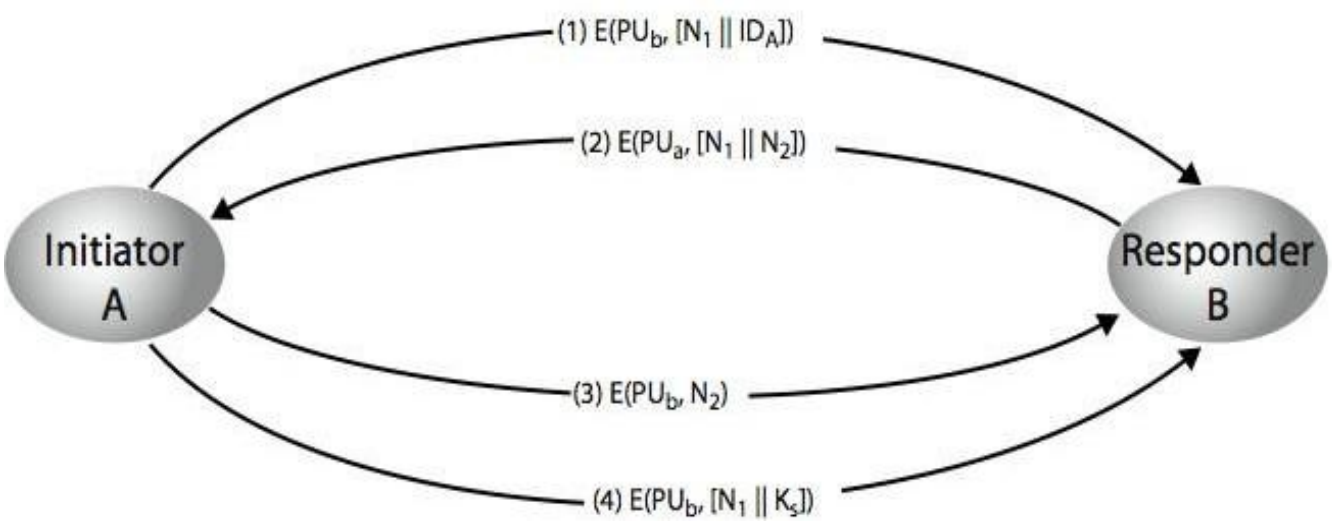
- Use previous methods to obtain public-key
- Can use for secrecy or authentication
- But public-key algorithms are slow
- So usually want to use private-key encryption to protect message contents hence need a session key
- Have several alternatives for negotiating a suitable session

# Simple Secret Key Distribution

- Proposed by Merkle in 1979
  - A generates a new temporary public key pair
  - A sends B the public key and their identity
  - B generates a session key K sends it to A encrypted using the supplied public key
  - A decrypts the session key and both use
- Problem is that an opponent can intercept and impersonate both halves of protocol

# Public-Key Distribution of Secret Keys

- if have securely exchanged public-keys:



# Hybrid Key Distribution

- Retain use of private-key KDC
- Shares secret master key with each user
- Distributes session key using master key
- Public-key used to distribute master keys
  - especially useful with widely distributed users
- Rationale
  - performance
  - backward compatibility

# Diffie-Hellman Key Exchange

- First public-key type scheme proposed
- By Diffie & Hellman in 1976 along with the exposition of public key concepts
  - note: now know that Williamson (UK CESG) secretly proposed the concept in 1970
- is a practical method for public exchange of a secret key
- used in a number of commercial products

# Diffie-Hellman Key Exchange

- A public-key distribution scheme
  - cannot be used to exchange an arbitrary message
  - rather it can establish a common key
  - known only to the two participants
- Value of key depends on the participants (and their private and public key information)
- Based on exponentiation in a finite (Galois) field (modulo a prime or a polynomial) - easy
- Security relies on the difficulty of computing discrete logarithms (similar to factoring) – hard

# Diffie-Hellman Setup

- All users agree on global parameters:
  - large prime integer or polynomial q
  - a being a primitive root mod q
- Each user (eg. A) generates their key
  - chooses a secret key (number): xA < q
  - compute their **public key**: $y = a^{x_A} \bmod q_A$
- Each user makes public that key $y_A$

# Diffie-Hellman Key Exchange

- shared session key for users A & B is $K_{AB}$:

$$K_{AB} = a^{x_A . x_B} \bmod q$$

$$= yA^{x_B} \bmod q \quad (\text{which B can compute})$$

$$= y_B^{x_A} \bmod q \quad (\text{which A can compute})$$

- $K_{AB}$ is used as session key in private-key encryption scheme between Alice and Bob
- if Alice and Bob subsequently communicate, they will have the **same** key as before, unless they choose new public-keys
- attacker needs an x, must solve discrete log

# Diffie-Hellman Example

- Users Alice & Bob who wish to swap keys:
- Agree on prime $q=353$ and $a=3$
- Select random secret keys:
  - A chooses xA=97, B chooses xB=233
- compute respective public keys:
  - $97$
- $y_A=3 \quad \textbf{mod 353 = 40} \quad$ (Alice)
- $\mathbf{y_B=3}^{233} \quad \textbf{mod 353 = 248} \quad$ (Bob)
- compute shared session key as:
- $K_{AB} = y_B^{x_A} \bmod 353 = 248^{97} = 160$ **(Alice)**
- $K_{AB} = y_A^{x_B} \bmod 353 = 40^{233} = 160$ **(Bob)**

# Key Exchange Protocols

- users could create random private/public D-H keys each time they communicate
- users could create a known private/public D-H key and publish in a directory, then consulted and used to securely communicate with them
- both of these are vulnerable to a meet-in-the-Middle Attack
- authentication of the keys is needed

# Elliptic Curve Cryptography

- majority of public-key crypto (RSA, D-H) use either integer or polynomial arithmetic with very large numbers/polynomials
- imposes a significant load in storing and processing keys and messages
- an alternative is to use elliptic curves
- offers same security with smaller bit sizes
- newer, but not as well analysed


# UNIT– III
# MESSAGE AUTHENTICATION ALGORITHM AND HASH FUNCTIONS

| CLOs | Course Learning Outcome |
|---|---|
| CLO1 | Explain the role of the message authentication in message transmission. |
| CLO2 | Explain the need of digital signature in message transmission. |
| CLO3 | Explain and demonstrate the role of different types of hash functions for providing security. |
| CLO4 | Understand the differences between the symmetric and symmetric cryptography algorithms for providing security. |

# Message Authentication Codes

- As shown the MAC provides authentication
- Can also use encryption for secrecy
  - generally use separate keys for each
  - can compute MAC either before or after encryption
  - is generally regarded as better done before
- Why use a MAC?
  - sometimes only authentication is needed
  - sometimes need authentication to persist longer than the encryption (eg. archival use)
- Note that a MAC is not a digital signature

# MAC Properties

- a MAC is a cryptographic checksum
  - MAC = CK(M)
  - condenses a variable-length message M
  - using a secret key K
  - to a fixed-sized authenticator
- is a many-to-one function
  - potentially many messages have same MAC
  - but finding these needs to be very difficult

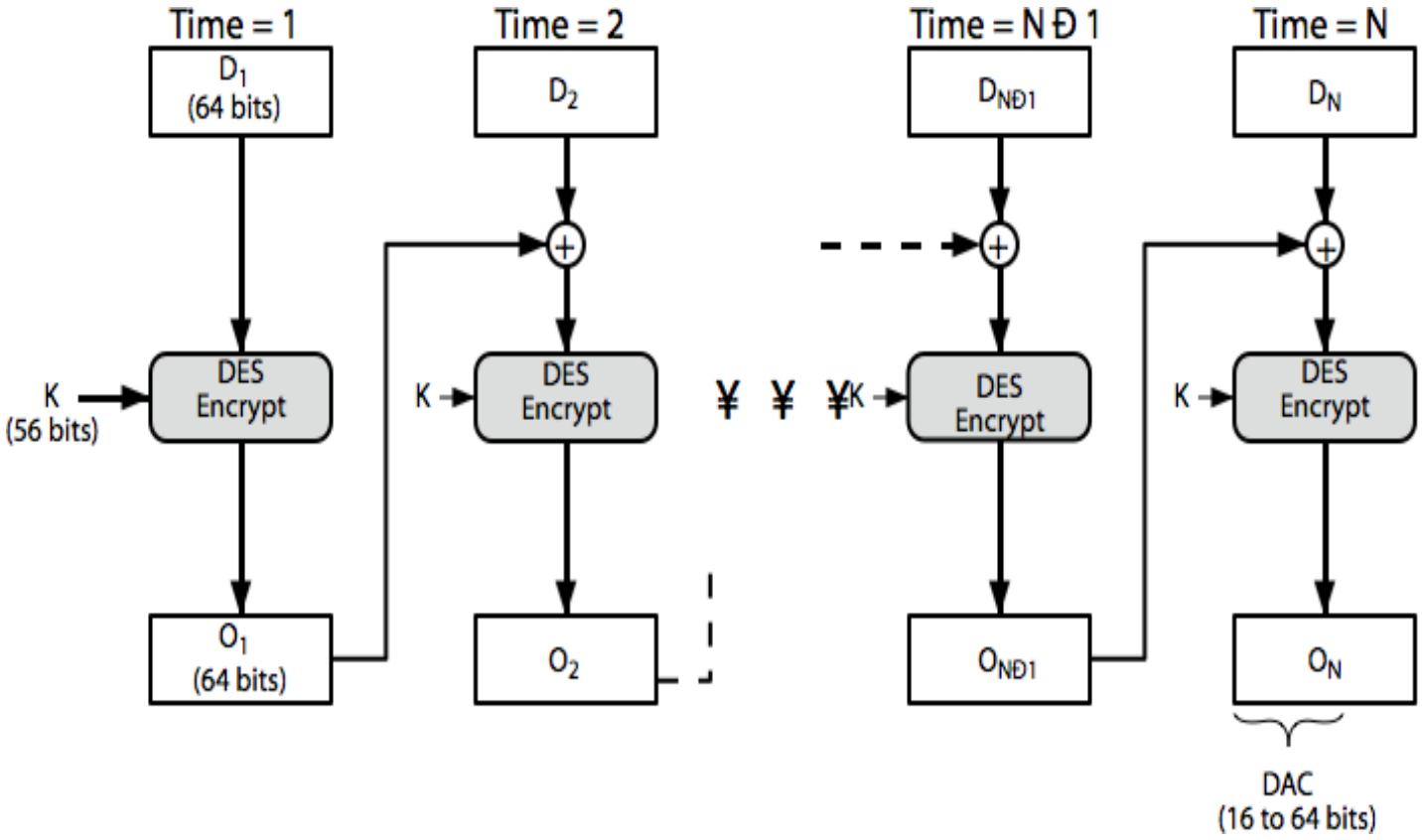
# Requirements for MACs

- Taking into account the types of attacks
- Need the MAC to satisfy the following:
  - knowing a message and MAC, is infeasible to find another message with same MAC
  - MACs should be uniformly distributed
  - MAC should depend equally on all bits of the message

# Using Symmetric Ciphers for MACs

- Can use any block cipher chaining mode and use final block as a MAC
- **Data Authentication Algorithm (DAA)** is a widely used MAC based on DES-CBC
  - using IV=0 and zero-pad of final block
  - encrypt message using DES in CBC mode
  - and send just the final block as the MAC
    or the leftmost M bits (16≤M≤64) of final block
- But final MAC is now too small for security

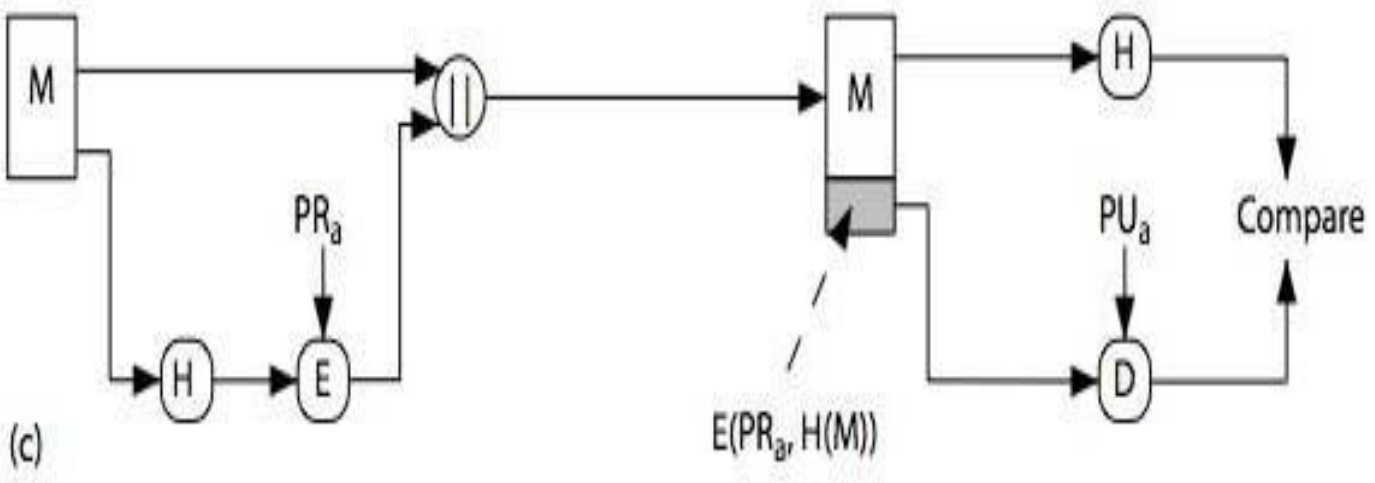
# Data Authentication Algorithm



# Hash Functions

- condenses arbitrary message to fixed size
  - h = H(M)
- usually assume that the hash function is public and not keyed
  - cf. MAC which is keyed
- hash used to detect changes to message
- can use in various ways with message
- most often to create a digital signature

# Hash Functions & Digital Signatures



# Requirements for Hash Functions

- Can be applied to any sized message M
- Produces fixed-length output h

  is easy to compute h=H(M)for any message M
- Given his infeasible to find xs.t. H(x)=h
  - one-way property
- Given xis infeasible to find ys.t. H(y)=H(x)
  - weak collision resistance
- Is infeasible to find any x,y s.t. H(y)=H(x)
  - strong collision resistance

# Simple Hash Functions

- There are several proposals for simple functions based on XOR of message blocks
- not secure since can manipulate any message and either not change hash or change hash also
- need a stronger cryptographic function (next chapter)

# Hash and MAC Algorithms

- Hash Functions
  - condense arbitrary size message to fixed size
  - by processing message in blocks
  - through some compression function
  - either custom or block cipher based
- Message Authentication Code (MAC)
  - fixed sized authenticator for some message
  - to provide authentication for message
  - by using block cipher mode or hash function

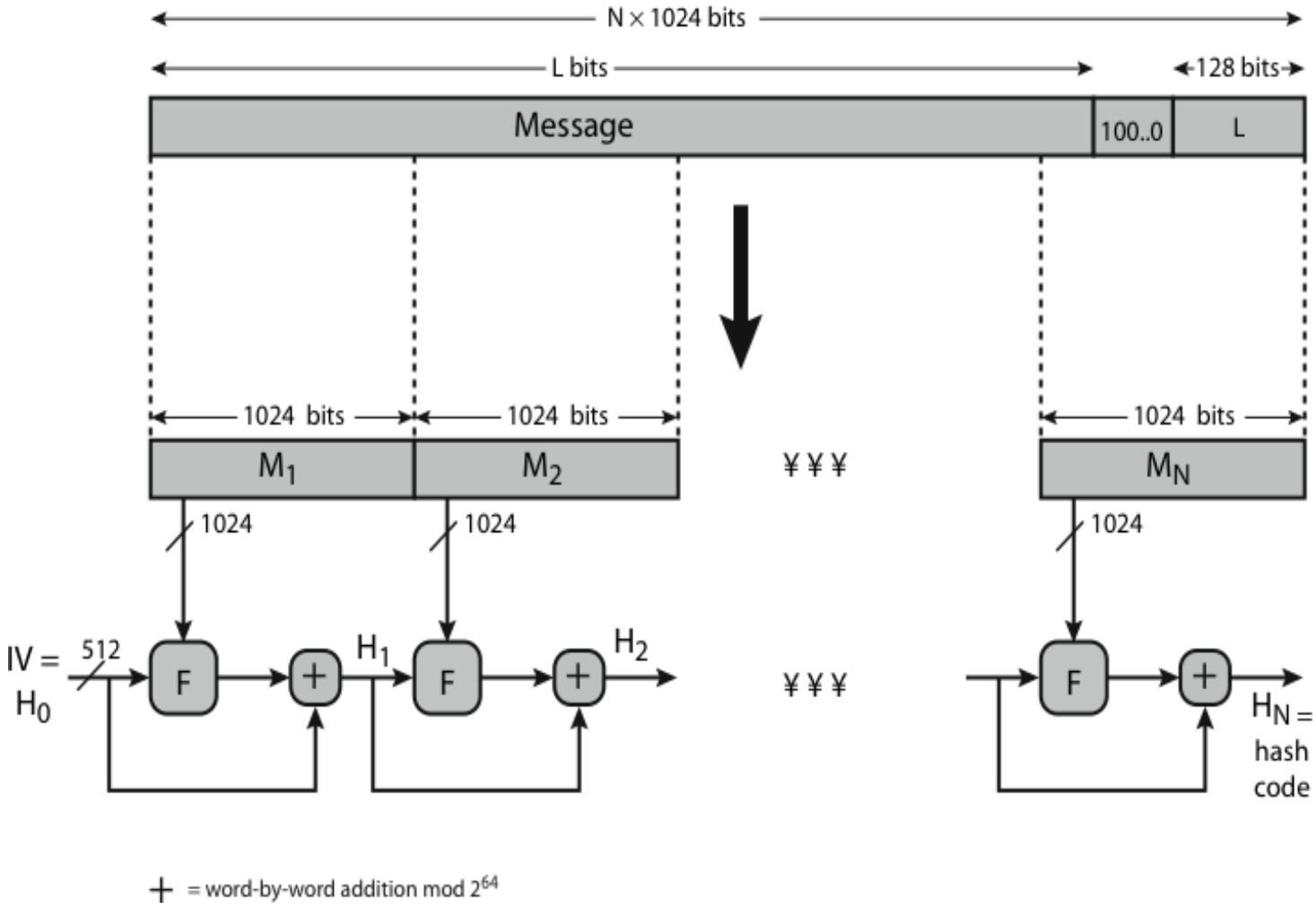
# Secure Hash Algorithm

- SHA originally designed by NIST & NSA in 1993
- was revised in 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
  - nb. the algorithm is SHA, the standard is SHS
- based on design of MD4 with key differences
- produces 160-bit hash values
- recent 2005 results on security of SHA-1 have raised concerns on its use in future applications

# Revised Secure Hash Standard

- NIST issued revision FIPS 180-2 in 2002
- Adds 3 additional versions of SHA
  - SHA-256, SHA-384, SHA-512
- Designed for compatibility with increased security provided by the AES cipher
- Structure & detail is similar to SHA-1
- Hence analysis should be similar
- But security levels are rather higher

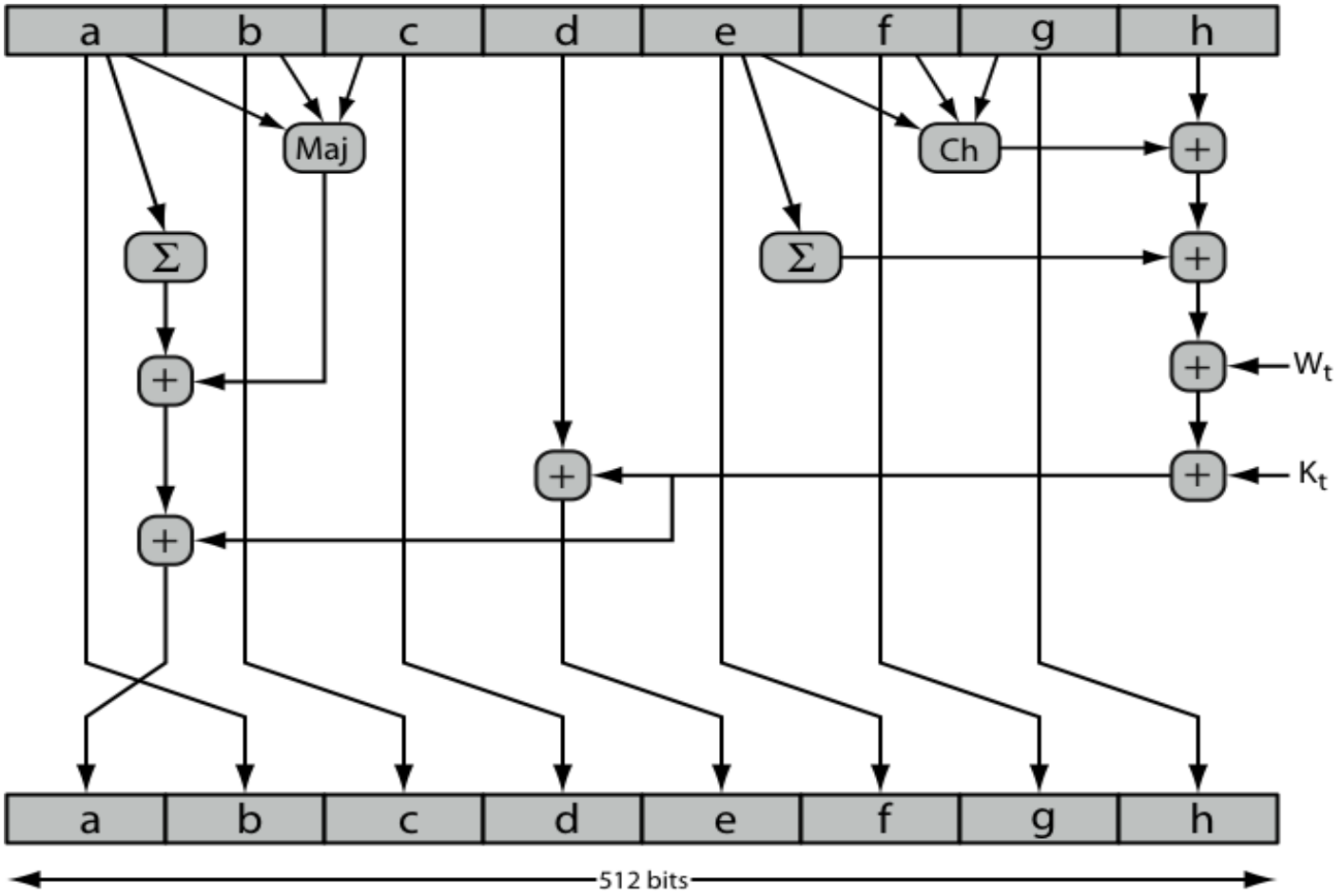
# SHA-512 Overview



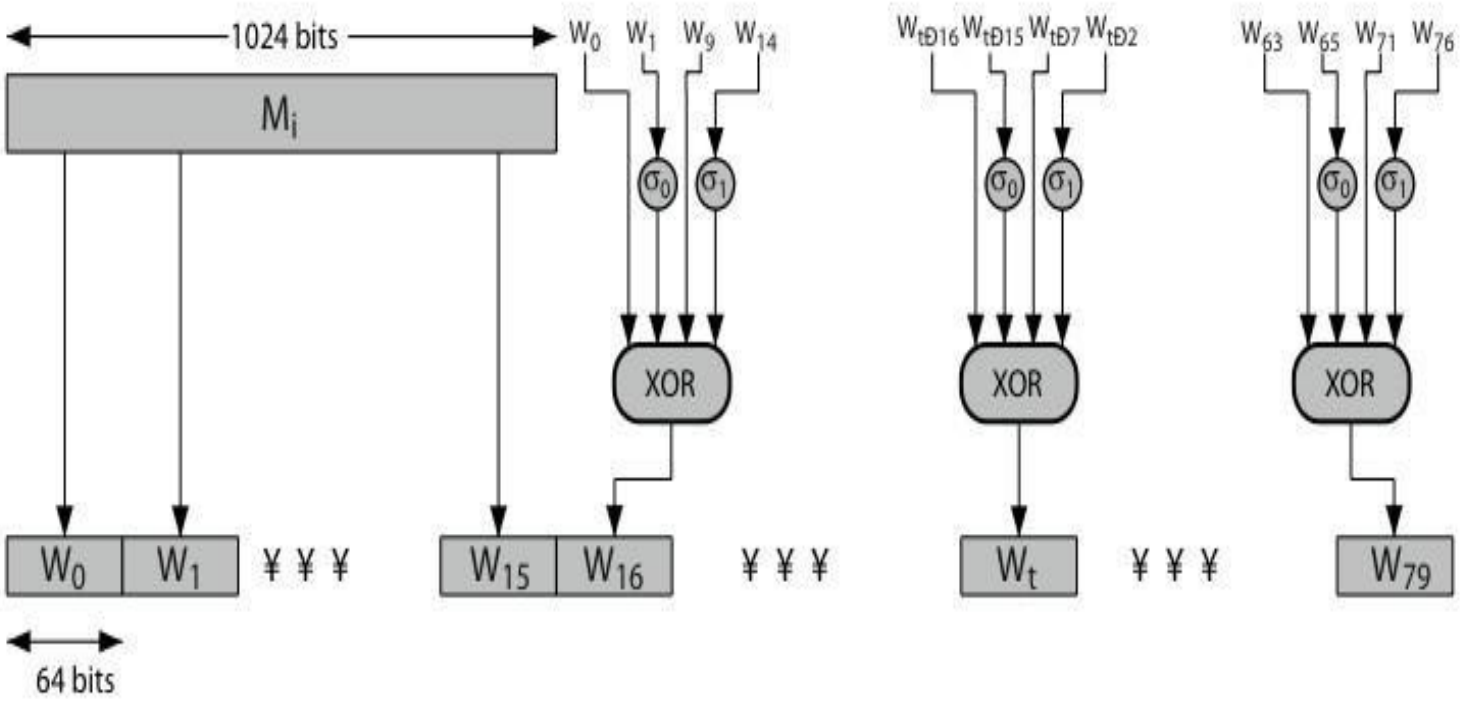
# SHA-512 Compression Function

- Heart of the algorithm
- Processing message in 1024-bit blocks
- Consists of 80 rounds
  - updating a 512-bit buffer
  - using a 64-bit value Wt derived from the current message block
  - and a round constant based on cube root of first 80 prime numbers

# SHA-512 Round Function



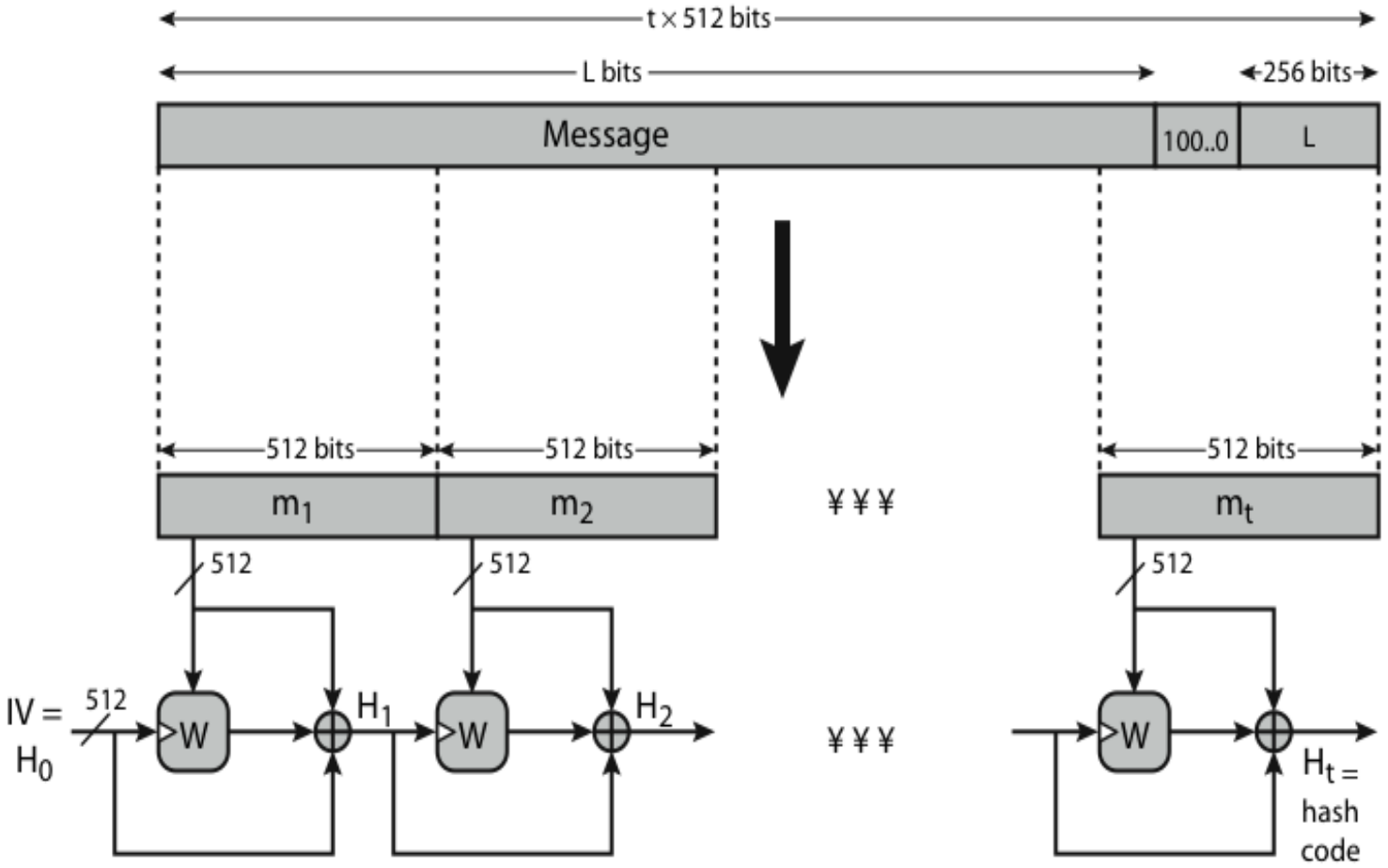
# SHA-512 Round Function



# Whirlpool

- Now examine the Whirlpool hash function
- Endorsed by European NESSIE project
- Uses modified AES internals as compression function
- Addressing concerns on use of block ciphers seen previously
- With performance comparable to dedicated algorithms like SHA

# Whirlpool Overview

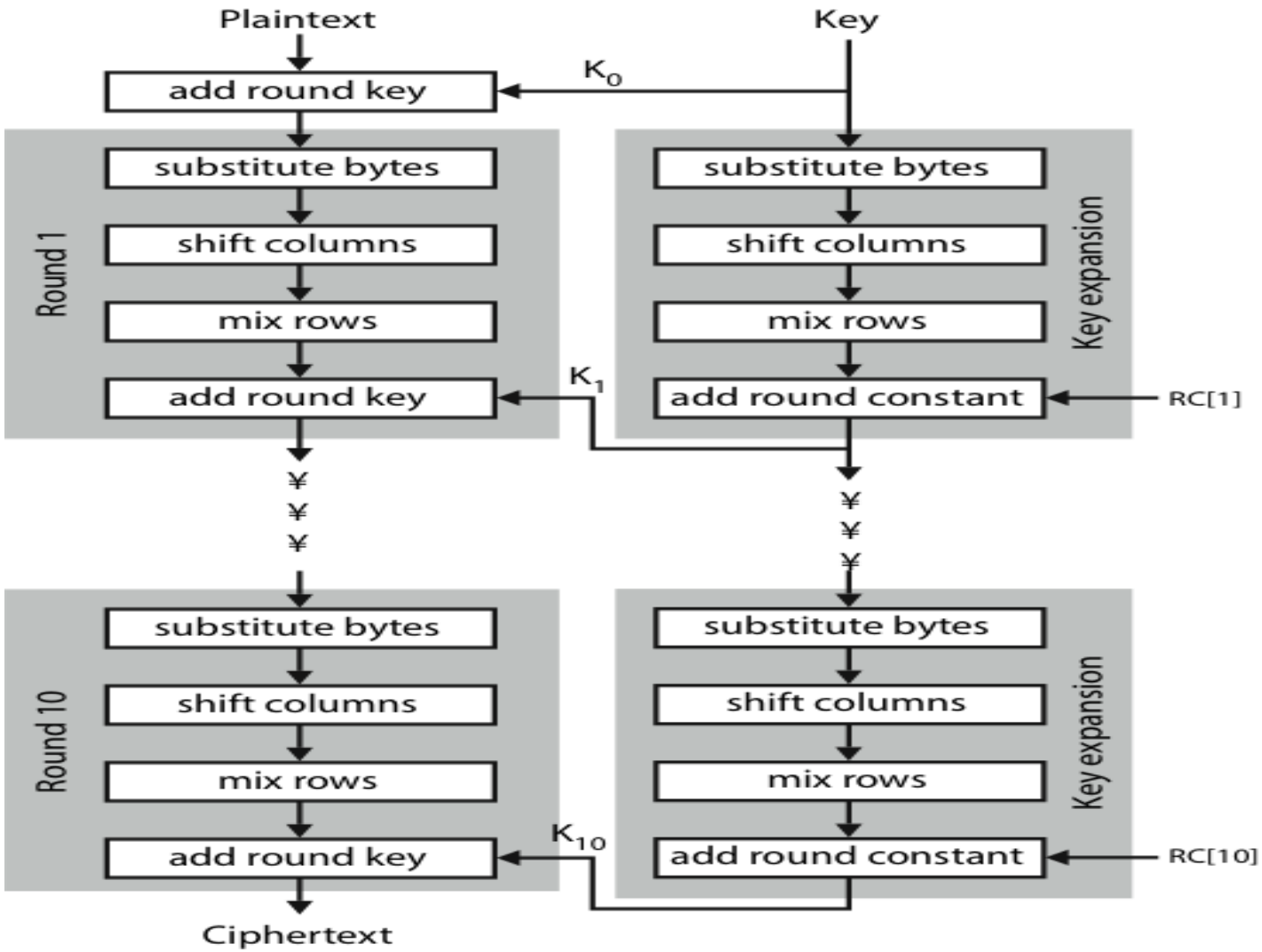



Note: triangular hatch marks key input

# Whirlpool Block Cipher

- Designed specifically for hash function use
- With security and efficiency of AES
- But with 512-bit block size and hence hash
- Similar structure & functions as AES but
  - input is mapped row wise
  - has 10 rounds
  - a different primitive polynomial for GF(2^8)
  - uses different S-box design & values

# Whirlpool Block Cipher



# Whirlpool Performance & Security

- Whirlpool is a very new proposal
- Hence little experience with use
- But many AES findings should apply
- Does seem to need more h/w than SHA, but with better resulting performance

# Keyed Hash Functions as MACs

- Want a MAC based on a hash function
  - because hash functions are generally faster
  - code for crypto hash functions widely available
- Hash includes a key along with message
- Original proposal:
  - KeyedHash = Hash(Key|Message)
  - some weaknesses were found with this
- Eventually led to development of HMAC

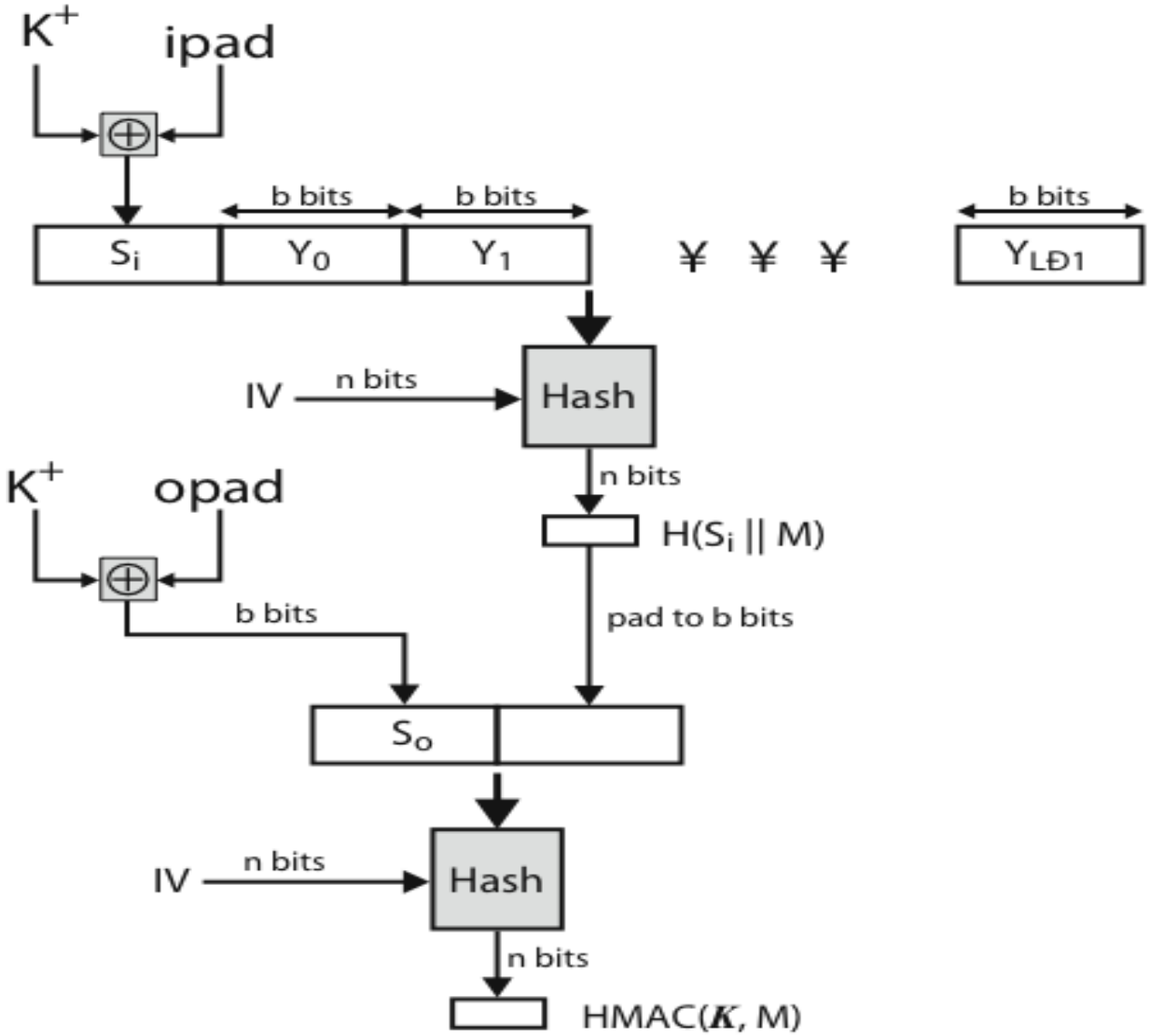
# HMAC

- Specified as Internet standard RFC2104
- Uses hash function on the message:

$$HMAC_K = Hash[(K^+ \text{ XOR } opad) || Hash[(K^+ \text{ XOR } ipad)||M)]]$$

- Where K+ is the key padded out to size
- And opad, ipad are specified padding constants
- Overhead is just 3 more hash calculations than the message needs alone
- Any hash function can be used
  - eg. MD5, SHA-1, RIPEMD-160, Whirlpool

# HMAC Overview



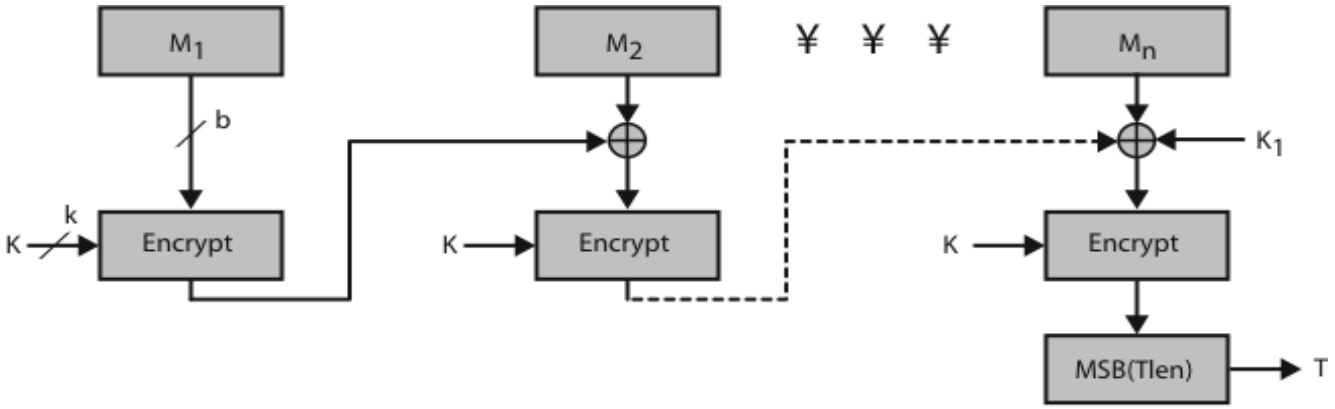
# HMAC Security

- Proved security of HMAC relates to that of the underlying hash algorithm
- Attacking HMAC requires either:
  - brute force attack on key used
  - birthday attack (but since keyed would need to observe a very large number of messages)
- Choose hash function used based on speed verses security constraints

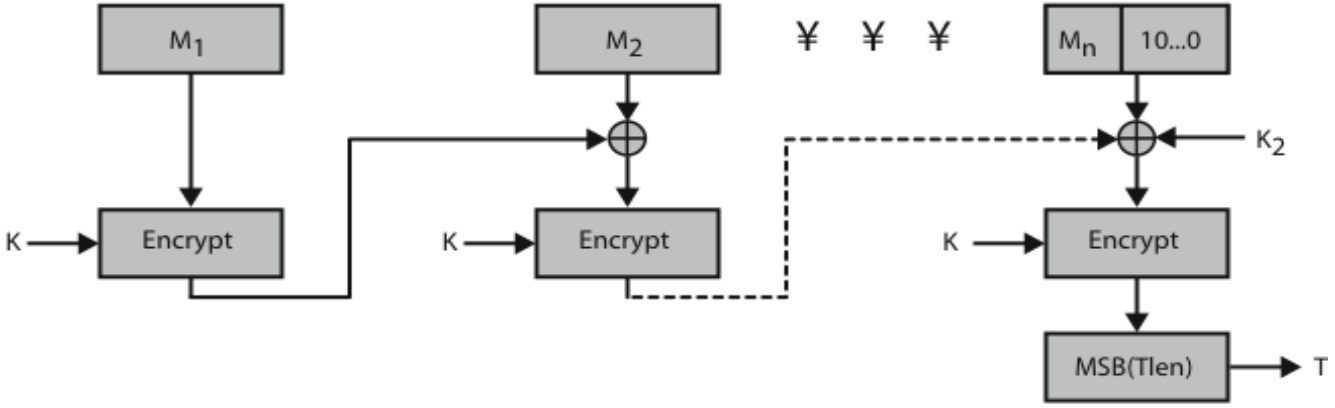
# CMAC

- Previously saw the DAA (CBC-MAC)
- Widely used in govt & industry
- But has message size limitation
- Can overcome using 2 keys & padding
- Thus forming the Cipher-based Message Authentication Code (CMAC)
- Adopted by NIST SP800-38B

# CMAC Overview




(a) Message length is integer multiple of block size




(b) Message length is not integer multiple of block size

Figure 12.12 Cipher-Based Message Authentication Code (CMAC)

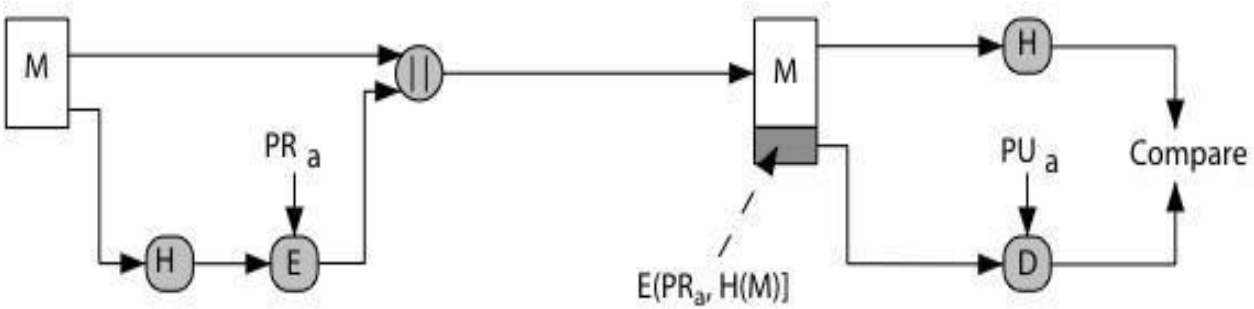
# Digital Signature Standard (DSS)

- US Govt approved signature scheme
- Designed by NIST & NSA in early 90's
- Published as FIPS-186 in 1991
- Revised in 1993, 1996 & then 2000
- Uses the SHA hash algorithm
- DSS is the standard, DSA is the algorithm
- FIPS 186-2 (2000) includes alternative RSA & elliptic curve signature variants

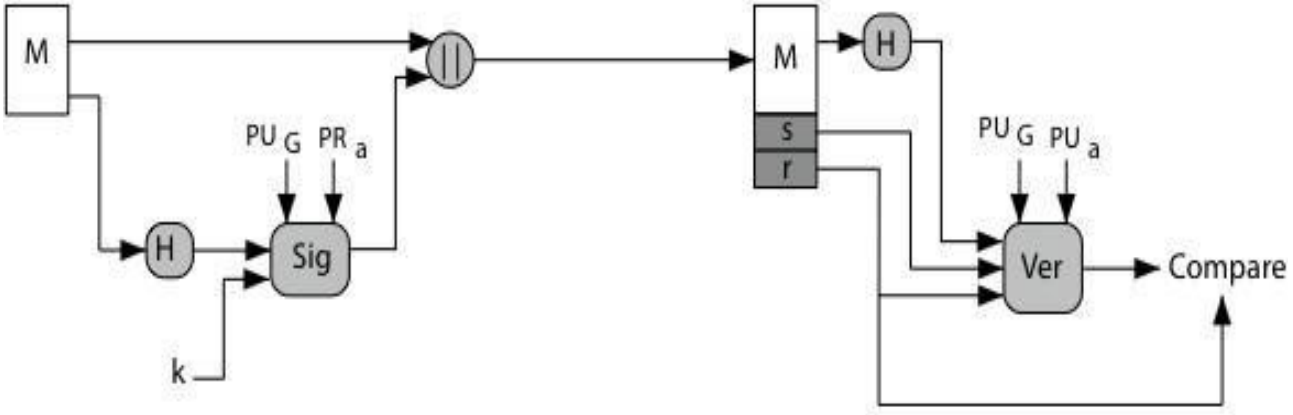
# Digital Signature Algorithm (DSA)

- Creates a 320 bit signature
- With 512-1024 bit security
- Smaller and faster than RSA
- A digital signature scheme only
- Security depends on difficulty of computing discrete logarithms
- Variant of ElGamal & Schnorr schemes

# Digital Signature Algorithm (DSA)




(a) RSA Approach




(b) DSS Approach

# DSA Key Generation

- Have shared global public key values (p,q,g):
  - choose q, a 160 bit
  - choose a large prime p = 2L
    - where L= 512 to 1024 bits and is a multiple of 64
    - and q is a prime factor of (p-1)
  - choose g = h(p-1)/q
    - where h<p-1, h(p-1)/q (mod p) > 1
- Users choose private & compute public key:
  - choose x<q
  - compute y = gx (mod p)

# DSA Signature Creation

- To **sign** a message Mthe sender:
  - generates a random signature key k, k<q
  - nb. k must be random, be destroyed after use, and never be reused
- Then computes signature pair:
  - r = (gk(mod p))(mod q)
  - s = (k-1.H(M)+ x.r)(mod q)
- Sends signature (r,s) with message M

# DSA Signature Verification

- Having received M & signature (r,s)
- To **verify** a signature, recipient computes:
  - w = s-1(mod q)
  - u1= (H(M).w)(mod q)
  - u2= (r.w)(mod q)
  - v = (gu1.yu2(mod p)) (mod q)
- If v=rthen signature is verified
- see book web site for details of proof why

# Kerberos

- trusted key server system from MIT
- provides centralised private-key third-party authentication in a distributed network
  - allows users access to services distributed through network
  - without needing to trust all workstations
  - rather all trust a central authentication server
- two versions in use: 4 & 5

# Kerberos Requirements

- Its first report identified requirements as:
  - secure
  - reliable
  - transparent
  - scalable
- Implemented using an authentication protocol based on Needham-Schroeder

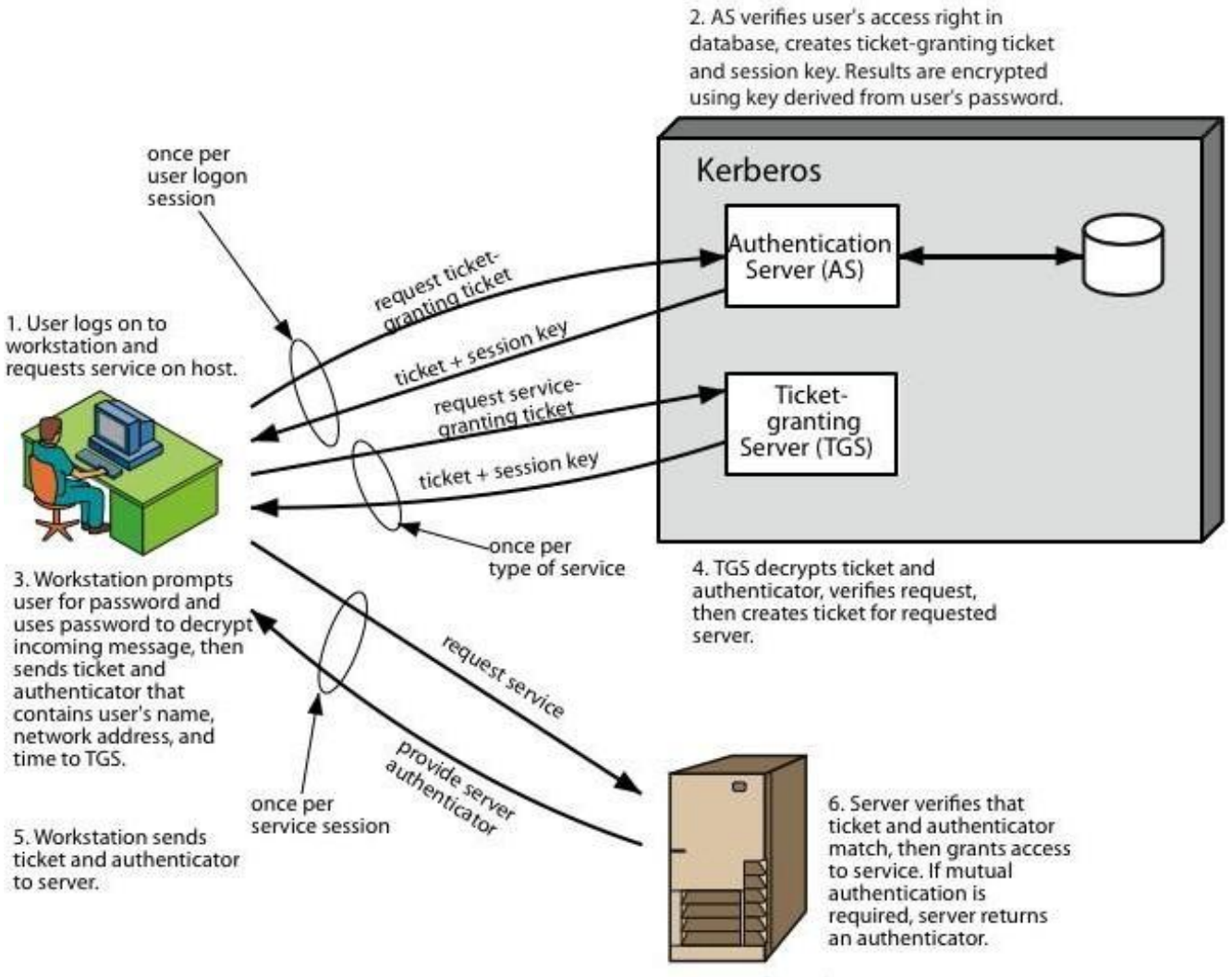
# Kerberos v4 Overview

- A basic third-party authentication scheme
- Have an Authentication Server (AS)
  - users initially negotiate with AS to identify self
  - AS provides a non-corruptible authentication credential (ticketgranting ticket TGT)
- Have a Ticket Granting server (TGS)
  - users subsequently request access to other services from TGS on basis of users TGT

# Kerberos v4 Dialogue

- Obtain ticket granting ticket from AS
  - once per session
- Obtain service granting ticket from TGT
  - for each distinct service required
- Client/server exchange to obtain service
  - on every service request

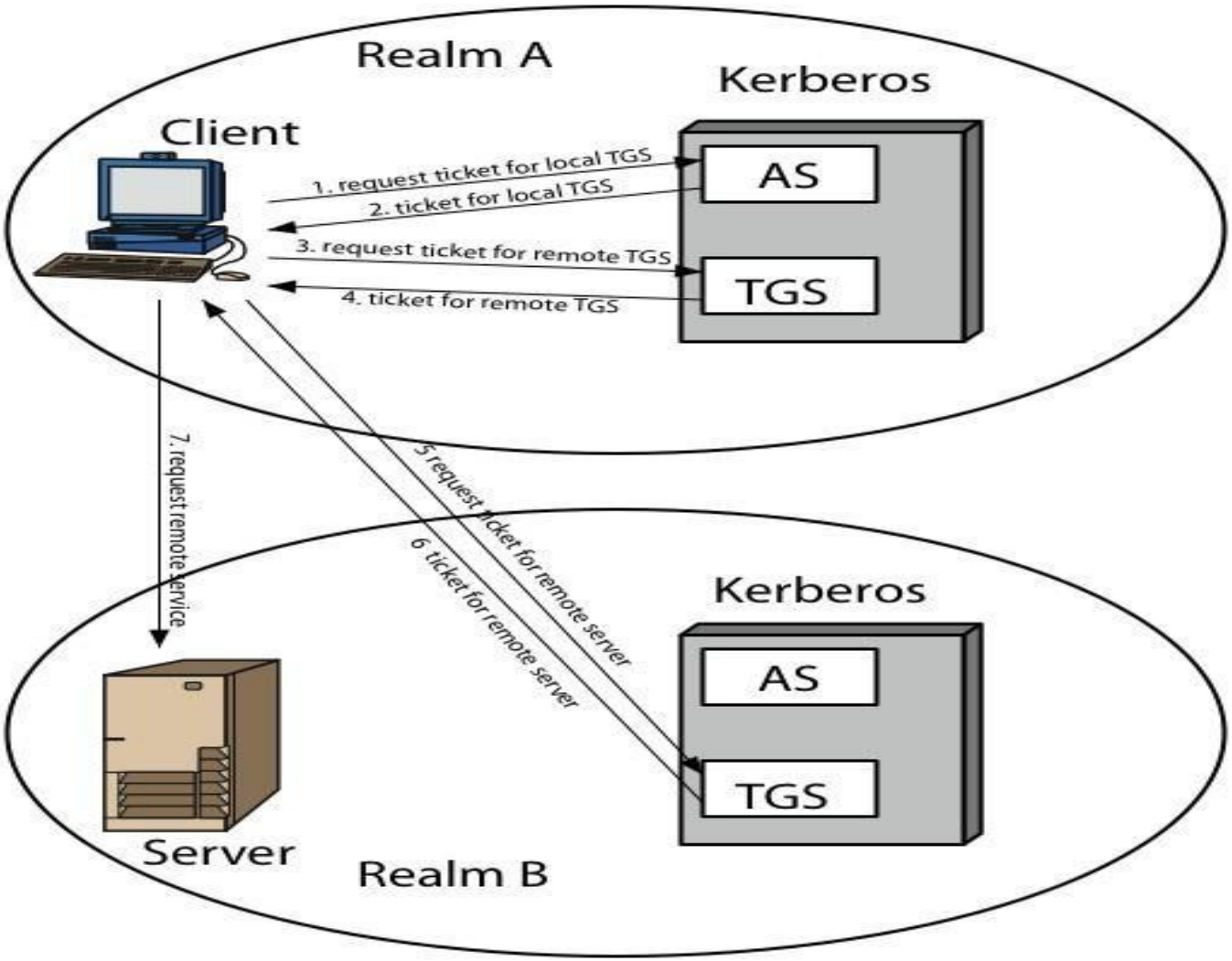
# Kerberos 4 Overview



# Kerberos Realms

- A Kerberos environment consists of:
  - a Kerberos server
  - a number of clients, all registered with server
  - application servers, sharing keys with server
- This is termed a realm
  - typically a single administrative domain
- If have multiple realms, their Kerberos servers must share keys and trust

# Kerberos Realms



# Kerberos Version 5

- Developed in mid 1990's
- Specified as Internet standard RFC 1510
- Provides improvements over v4
  - Addresses environmental shortcomings
    - Encryption alg, network protocol, byte order, ticket lifetime,
      - authentication forwarding, interrealm auth
  - And technical deficiencies
    - double encryption, non-std mode of use, session keys, password attacks

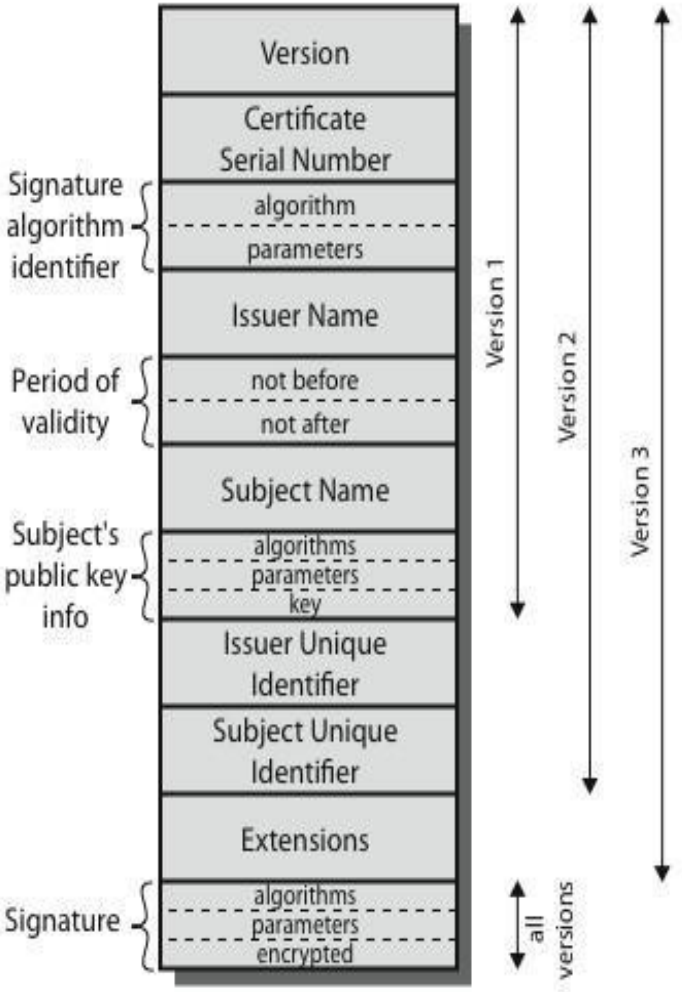
# X.509 Authentication Service

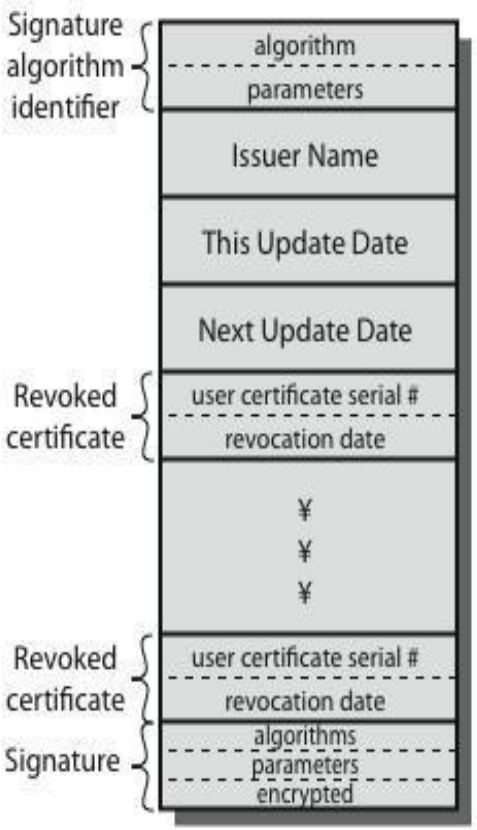
- Part of CCITT X.500 directory service standards
  - Distributed servers maintaining user info database
- Defines framework for authentication services
  - directory may store public-key certificates
  - with public key of user signed by certification authorityalso defines authentication protocols
- uses public-key crypto & digital signatures
  - algorithms not standardised, but RSA recommended
- X.509 certificates are widely used

# X.509 Certificates

- Issued by a Certification Authority (CA), containing:
  - version (1, 2, or 3)serial number (unique within CA) identifying certificate
  - signature algorithm identifier issuer X.500 name (CA)period of validity (from - to dates)
  - Subject X.500 name (name of owner)
  - Subject public-key info (algorithm, parameters, key),Issuer unique identifier (v2+),Subject unique identifier (v2+)
  - Extension fields (v3),Signature (of hash of all fields in certificate)
- Notation CA<<A>> denotes certificate for A signed by CA

# X.509 Certificates




(b) Certificate Revocation List

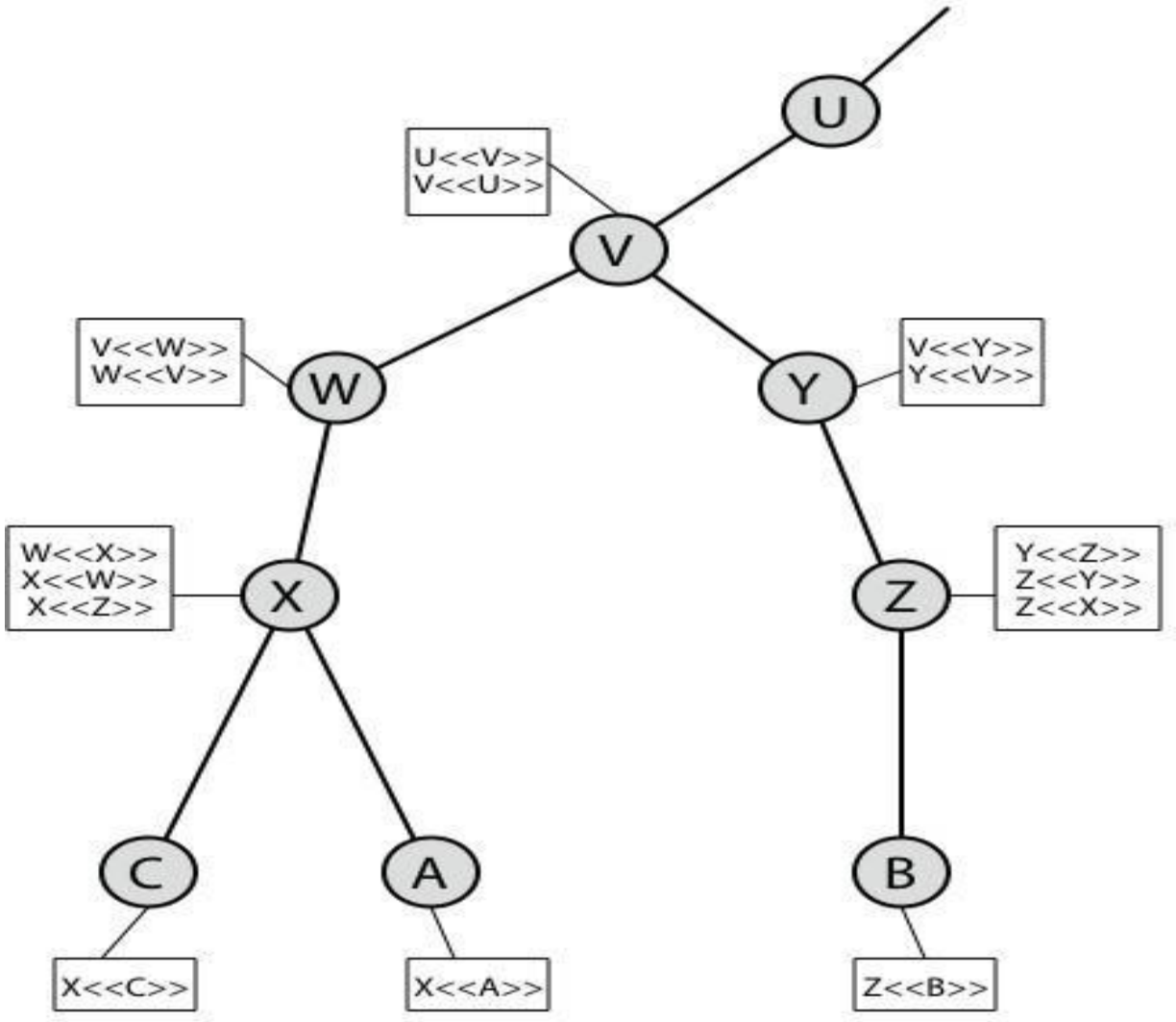
# Obtaining a Certificate

- Any user with access to CA can get any certificate from it
- Only the CA can modify a certificate
- Because cannot be forged, certificates can be placed in a public directory

# CA Hierarchy

- If both users share a common CA then they are assumed to know its public key
- Otherwise CA's must form a hierarchy
- Use certificates linking members of hierarchy to validate other CA's
  - Each CA has certificates for clients (forward) and parent (backward)
- Each client trusts parents certificates
- Enable verification of any certificate from one CA by users of all other CAs in hierarchy

# CA Hierarchy Use



# Certificate Revocation

- Certificates have a period of validity
- May need to revoke before expiry, eg:
  - user's private key is compromised
  - user is no longer certified by this CA
  - CA's certificate is compromised
- CA’s maintain list of revoked certificates
  - the Certificate Revocation List (CRL)
- Users should check certificates with CA’s CRL

# Authentication Procedures

- X.509 includes three alternative authentication procedures:
- One-Way Authentication
- Two-Way Authentication
- Three-Way Authentication
- all use public-key signatures

# One-Way Authentication

- 1 message ( A->B) used to establish
  - the identity of A and that message is from A
  - message was intended for B
  - integrity & originality of message
- message must include timestamp, nonce, B's identity and is signed by A
- may include additional info for B
  - eg session key

# Two-Way Authentication

- 2 messages (A->B, B->A) which also establishes in addition:
  - the identity of B and that reply is from B
  - that reply is intended for A
  - integrity & originality of reply
- Reply includes original nonce from A, also timestamp  and nonce from B
- May include additional info for A

# Three-Way Authentication

- 3 messages (A->B, B->A, A->B) which enables above authentication without synchronized clocks
- has reply from A back to B containing signed copy of nonce from B
- means that timestamps need not be checked or relied upon

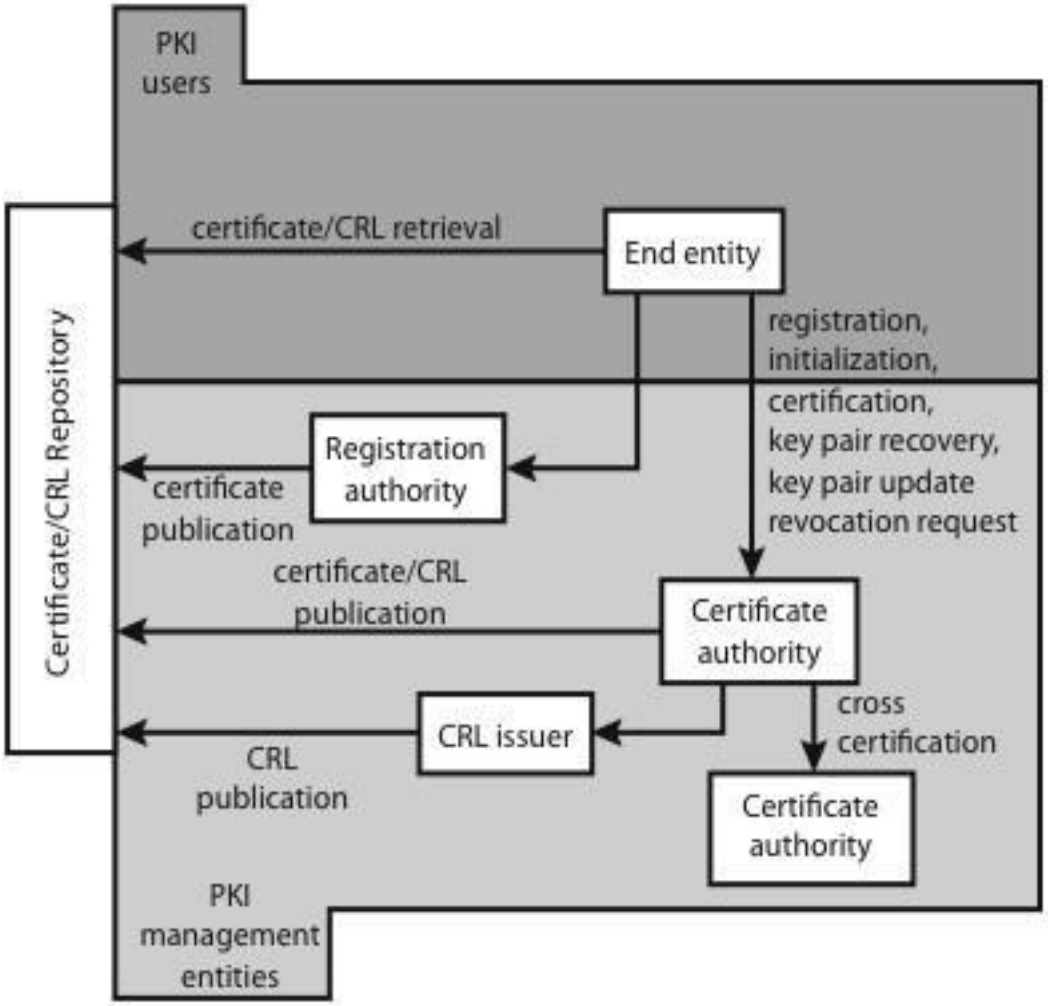
# X.509 Version 3

- Has been recognised that additional information is needed in a certificate
  - email/URL, policy details, usage constraints
- Rather than explicitly naming new fields defined a general extension method
- Extensions consist of:
  - extension identifier
  - criticality indicator
  - extension value

# Certificate Extensions

- Key and policy information
  - convey info about subject & issuer keys, plus indicators of certificate policy
- Certificate subject and issuer attributes
  - support alternative names, in alternative formats for certificate
    - subject and/or issuer
- Certificate path constraints
  - allow constraints on use of certificates by other CA's

# Public Key Infrastructure




# UNIT– IV
# E-MAIL SECURITY

| CLOs | Course Learning Outcome |
|---|---|
| CLO1 | Explain S/MIME and PGP for transmitting mail from sender to receiver. |
| CLO2 | Explain IP security for internet protocol and analyze how it provides security. |

# Email Security

- Email is one of the most widely used and regarded network services
- Currently message contents are not secure
  - may be inspected either in transit
  - or by suitably privileged users on destination system

# Email Security Enhancements

- Confidentiality
  - protection from disclosure
- Authentication
  - of sender of message
- Message integrity
  - protection from modification
- Non-repudiation of origin
  - protection from denial by sender

# Pretty Good Privacy (PGP)

- Widely used de facto secure email
- Developed by Phil Zimmermann
- Selected best available crypto algs to use
- Integrated into a single program
- On Unix, PC, Macintosh and other systems
- Originally free, now also have commercial versions available

# PGP Operation – Authentication

- sender creates message
- use SHA-1 to generate 160-bit hash of message
- signed hash with RSA using sender's private key, and is attached to message
- receiver uses RSA with sender's public key to decrypt and recover hash code
- receiver verifies received message using hash of it and compares with decrypted hash code

# PGP Operation – Confidentiality

- Sender generates message and 128-bit random number as session key for it
- Encrypt message using CAST-128 / IDEA / 3DES in CBC mode with session key
- Session key encrypted using RSA with recipient's public key, & attached to msg
- Receiver uses RSA with private key to decrypt and recover session key
- Session key is used to decrypt message

# PGP Operation – Confidentiality & Authentication

- can use both services on same message
  - Create signature & attach to message
  - Encrypt both message & signature
  - Attach RSA/ElGamal encrypted session key

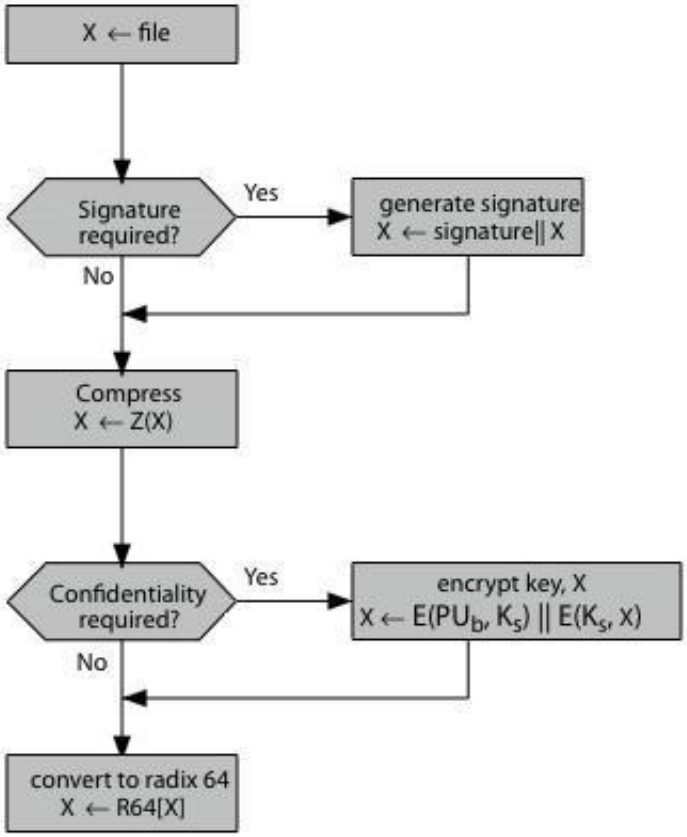
# PGP Operation – Compression

- By default PGP compresses message after signing but before encrypting
  - so can store uncompressed message & signature for later verification
  - & because compression is non deterministic
- Uses ZIP compression algorithm

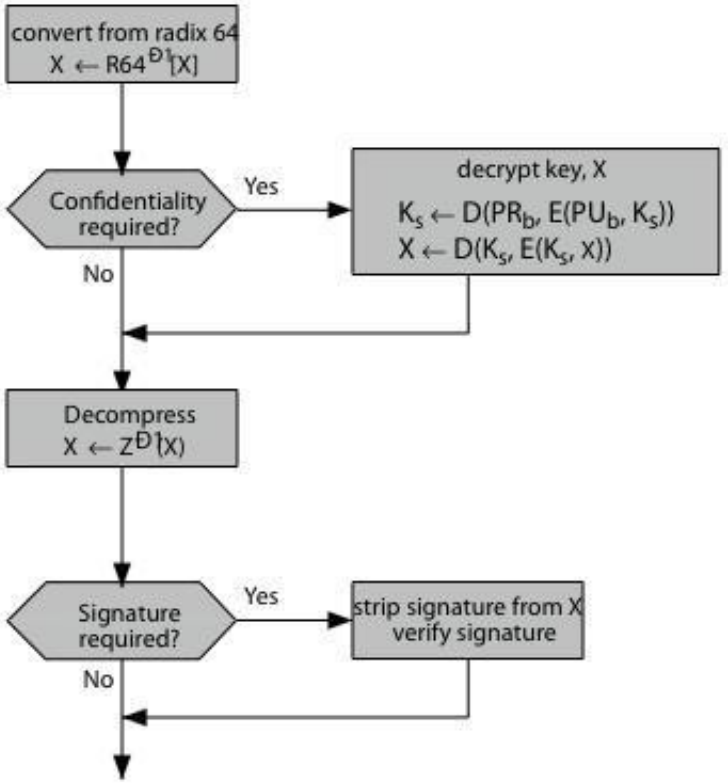
# PGP Operation – Email Compatibility

- When using PGP will have binary data to send (encrypted message etc)
- However email was designed only for text
- Hence PGP must encode raw binary data into printable ASCII characters
- Uses radix-64 algorithm
  - maps 3 bytes to 4 printable chars
  - also appends a CRC
- PGP also segments messages if too big

# PGP Operation – Summary




(a) Generic Transmission Diagram (from A)




(b) Generic Reception Diagram (to B)

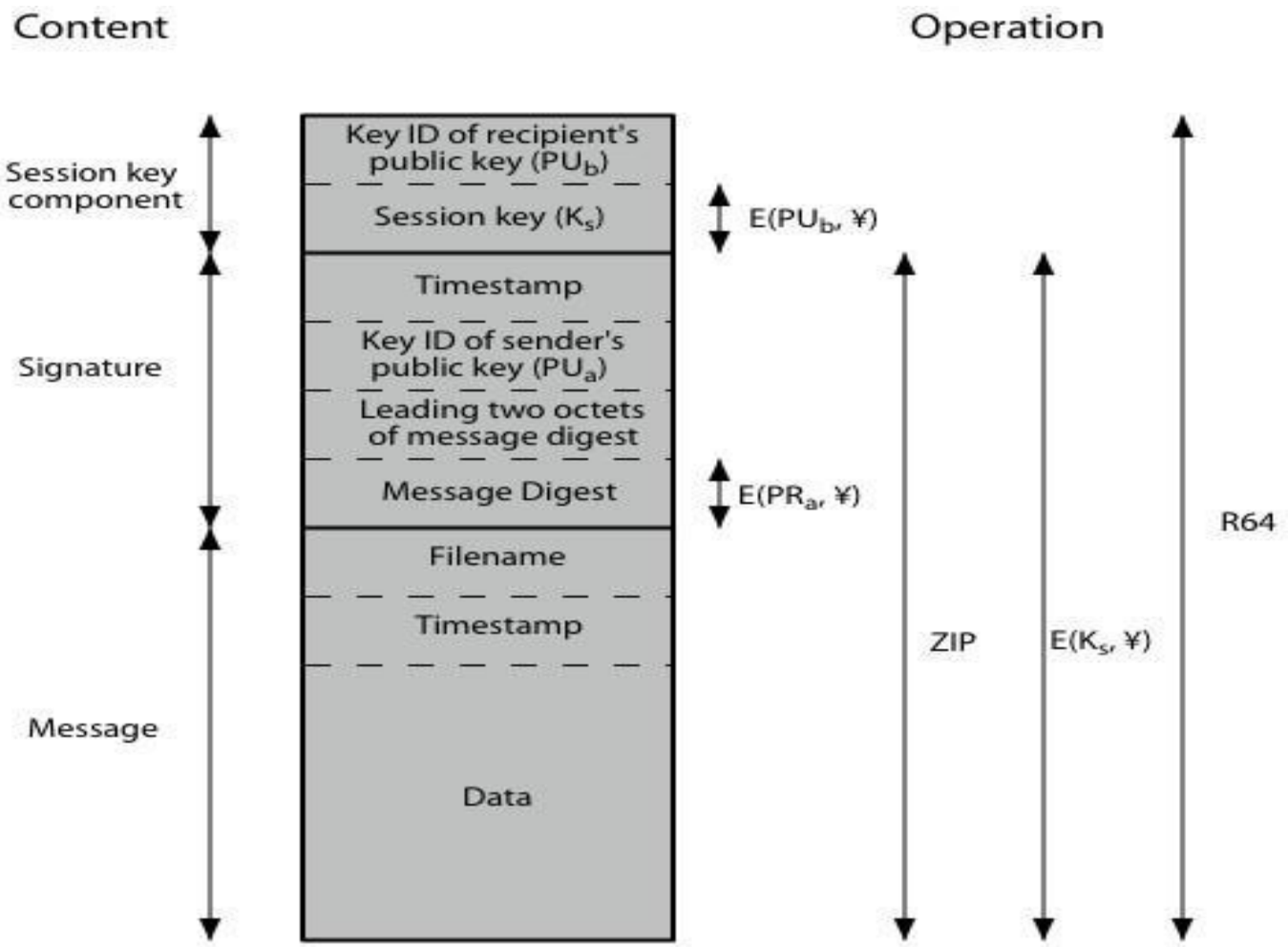
# PGP Session Keys

- need a session key for each message
  - of varying sizes: 56-bit DES, 128-bit CAST or IDEA, 168-bit Triple-DES
- generated using ANSI X12.17 mode
- uses random inputs taken from previous uses and from keystroke timing of user

# PGP Public & Private Keys

- since many public/private keys may be in use, need to identify which is actually used to encrypt session key in a message
  - could send full public-key with every message
  - but this is inefficient
- rather use a key identifier based on key
  - is least significant 64-bits of the key
  - will very likely be unique
- also use key ID in signatures

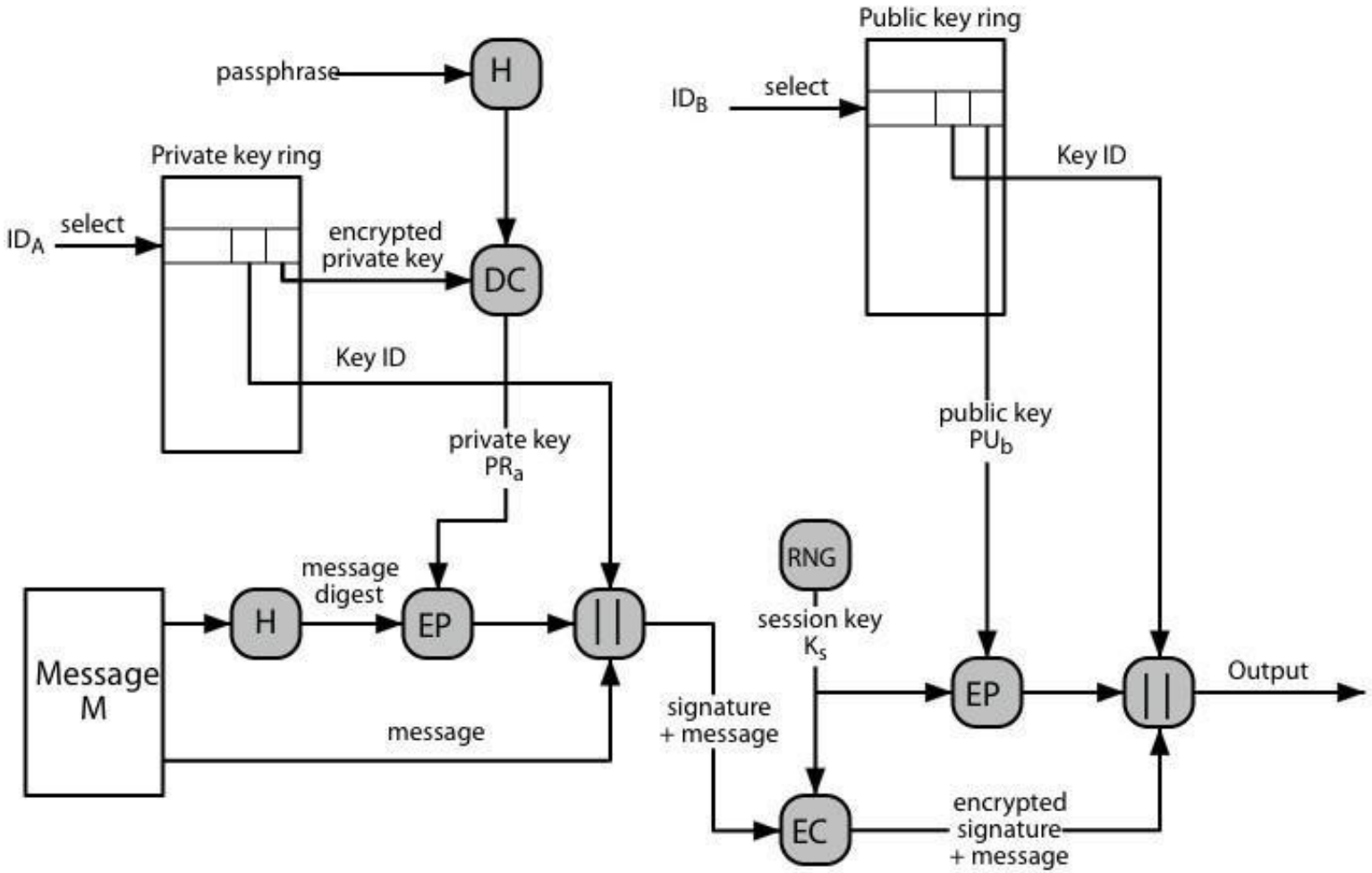
# PGP Message Format



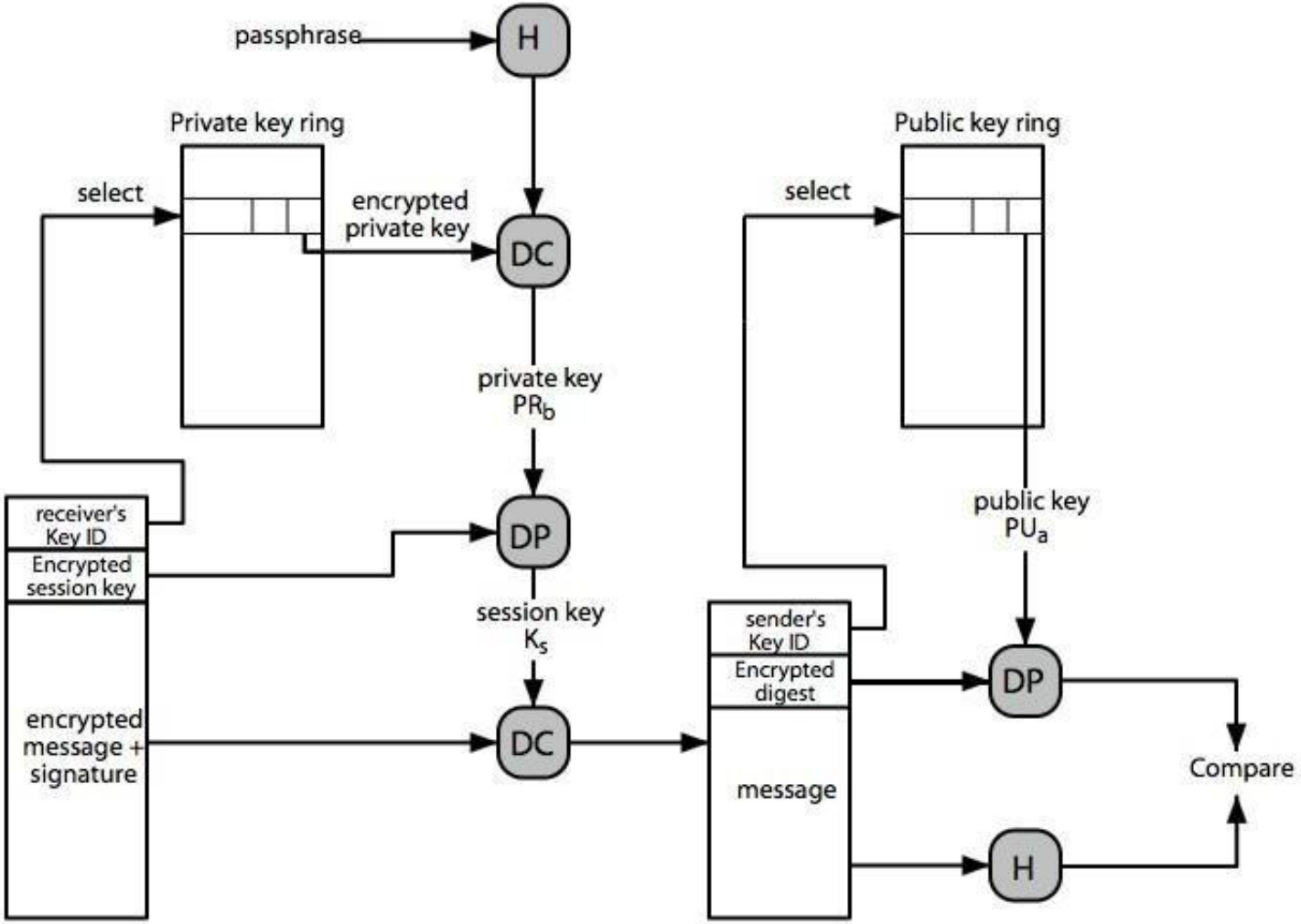
# PGP Key Rings

- Each PGP user has a pair of keyrings:
  - public-key ring contains all the public-keys of other PGP users
    - known to this user, indexed by key ID
  - private-key ring contains the public/private key pair(s) for this user, indexed by key ID & encrypted keyed from a hashed passphrase
- Security of private keys thus depends on the pass- phrase security

# PGP Message Generation



# PGP Message Reception



# PGP Key Management

- Rather than relying on certificate authorities
- In PGP every user is own CA
  - can sign keys for users they know directly
- Forms a “web of trust”
  - trust keys have signed
  - can trust keys others have signed if have a chain of signatures to them
- key ring includes trust indicators
- Users can also revoke their keys

# S/MIME (Secure/Multipurpose Internet Mail Extension)

- Security enhancement to MIME email
  - original Internet RFC822 email was text only
  - MIME provided support for varying content types and multi- part messages
  - with encoding of binary data to textual form
  - S/MIME added security enhancements
- Have S/MIME support in many mail agents
  - eg MS Outlook, Mozilla, Mac Mail etc

# S/MIME Functions

- Enveloped data
  - encrypted content and associated keys
- Signed data
  - encoded message + signed digest
- Clear-signed data
  - cleartext message + encoded signed digest
- Signed & enveloped data
  - nesting of signed & encrypted entities

# S/MIME Cryptographic Algorithms

- Digital signatures: DSS & RSA
- Hash functions: SHA-1 & MD5
- Session key encryption: ElGamal & RSA
- Message encryption: AES, Triple-DES, RC2/40 and others
- MAC: HMAC with SHA-1
- Have process to decide which algs to use

# S/MIME Messages

- S/MIME secures a MIME entity with a signature, encryption, or both
- Forming a MIME wrapped PKCS object
- Have a range of content-types:
  - enveloped data
  - signed data
  - clear-signed data
  - registration request
  - certificate only message

# S/MIME Certificate Processing

- S/MIME uses X.509 v3 certificates
- Managed using a hybrid of a strict X.509 CA hierarchy & PGP's web of trust
- Each client has a list of trusted CA's certs
- And own public/private key pairs & certs
- Certificates must be signed by trusted CA's

# Certificate Authorities

- Have several well-known CA's
- Verisign one of most widely used
- Verisign issues several types of Digital IDs
- Increasing levels of checks & hence trust

| Class | Identity Checks | Usage |
|---|---|---|
| 1 | name/email check | web browsing/email |
| 2 | + enroll/addr check | email, subs, s/w validate |
| 3 | + ID documents | e-banking/service access |

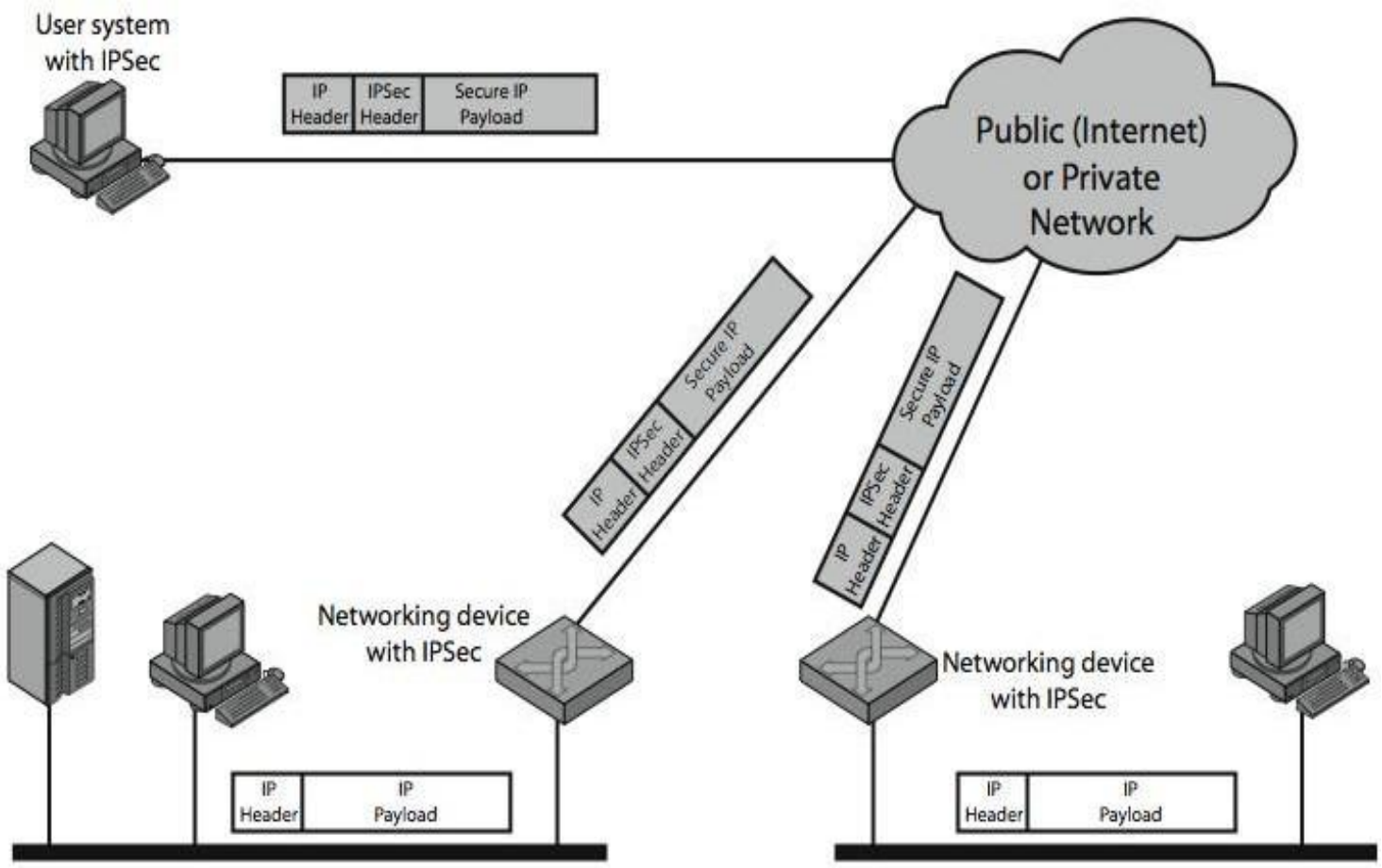
# IP Security

- Have a range of application specific security mechanisms
  - eg. S/MIME, PGP, Kerberos, SSL/HTTPS
- However there are security concerns that cut across protocol layers
- Would like security implemented by the network for all applications

# IPSec

- General IP Security mechanisms
- Provides
  - authentication
  - confidentiality
  - key management
- Applicable to use over LANs, across public & private WANs, & for the Internet

# IPSec Uses


User system with IPSec
IP Header
IPSec Header
Secure IP Payload
Public (Internet) or Private Network
IP Header
IPSec Header
Secure IP Payload
IP Header
IPSec Header
Secure IP Payload
Networking device with IPSec
Networking device with IPSec
IP Header
IP Payload
IP Header
IP Payload

# Benefits of IPSec

- In a firewall/router provides strong security to all traffic crossing the perimeter
- In a firewall/router is resistant to bypass
- Is below transport layer, hence transparent to applications
- Can be transparent to end users
- Can provide security for individual users
- Secures routing architecture

# IP Security Architecture

- Specification is quite complex
- Defined in numerous RFC's
  - incl. RFC 2401/2402/2406/2408
  - many others, grouped by category
- Mandatory in IPv6, optional in IPv4
- Have two security header extensions:
  - Authentication Header (AH)
  - Encapsulating Security Payload (ESP)

# IPSec Services

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets
  - A form of partial sequence integrity
- Confidentiality (encryption)
- Limited traffic flow confidentiality

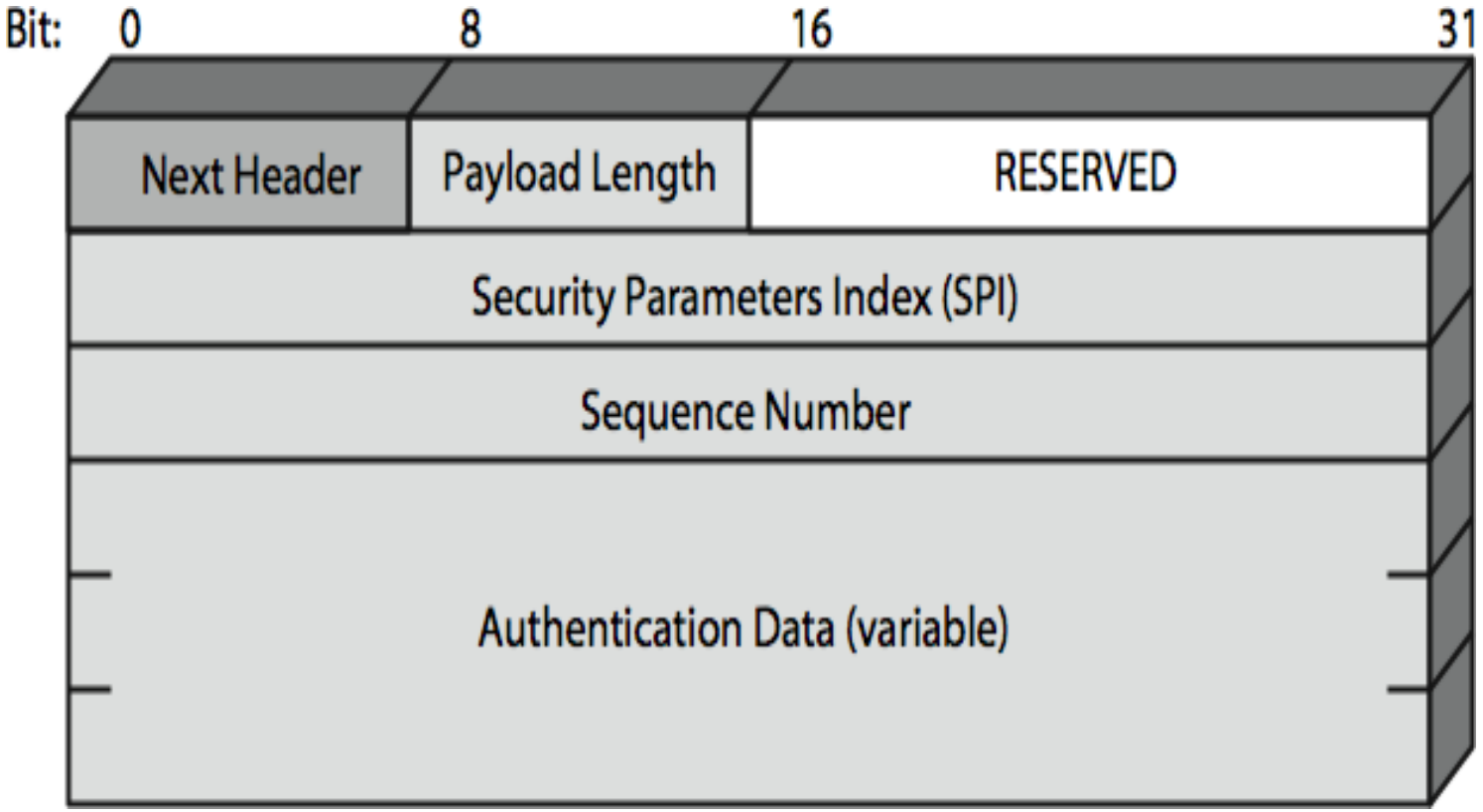
# Security Associations

- A one-way relationship between sender & receiver that affords security for traffic flow
- Defined by 3 parameters:
  - Security Parameters Index (SPI)
  - IP Destination Address
  - Security Protocol Identifier
- Has a number of other parameters
  - seq no, AH & EH info, lifetime etc
- Have a database of Security Associations

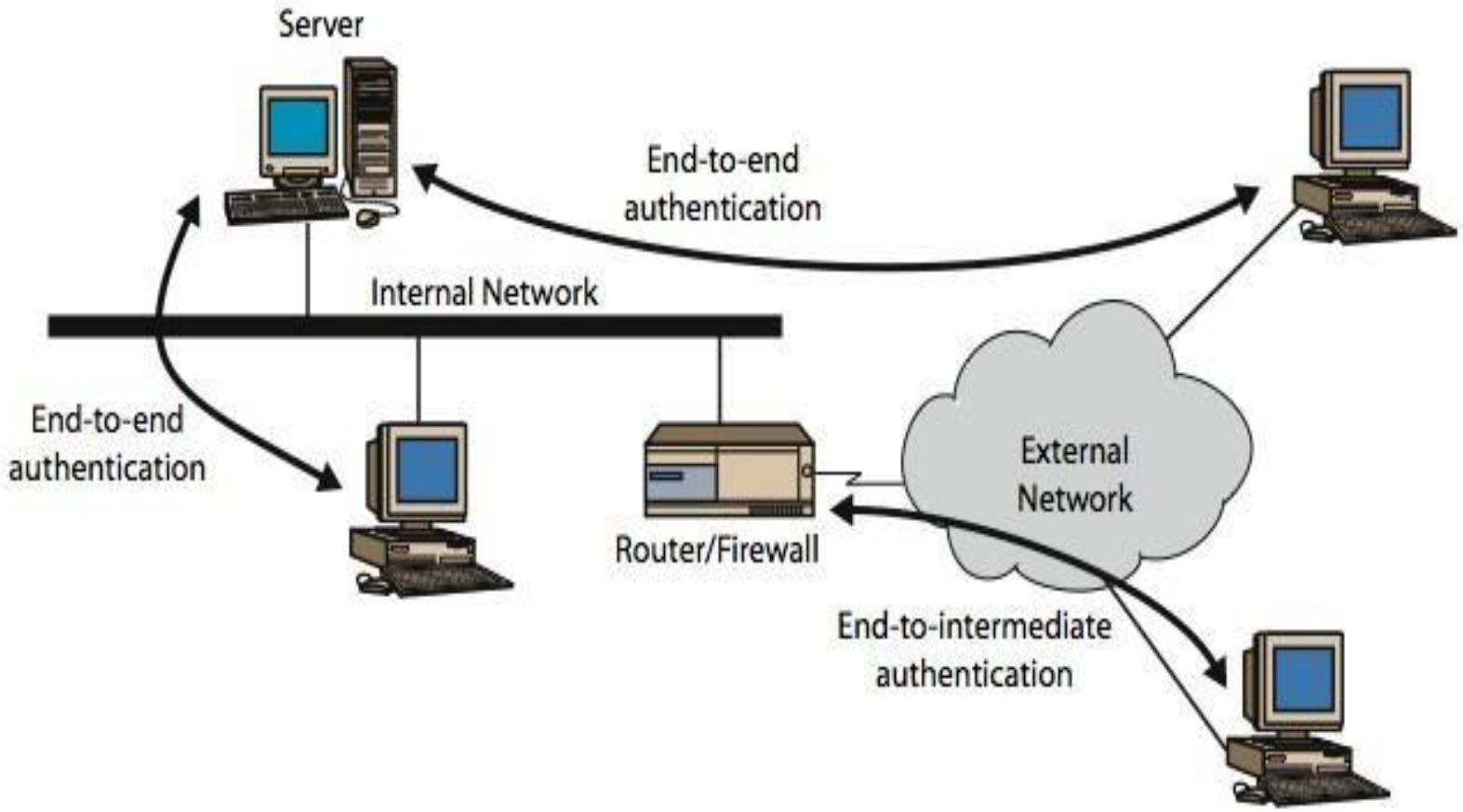
# Authentication Header (AH)

- Provides support for data integrity & authentication of IP packets
  - end system/router can authenticate user/app
  - prevents address spoofing attacks by tracking sequence numbers
- Based on use of a MAC
  - HMAC-MD5-96 or HMAC-SHA-1-96
- Parties must share a secret key

# Authentication Header



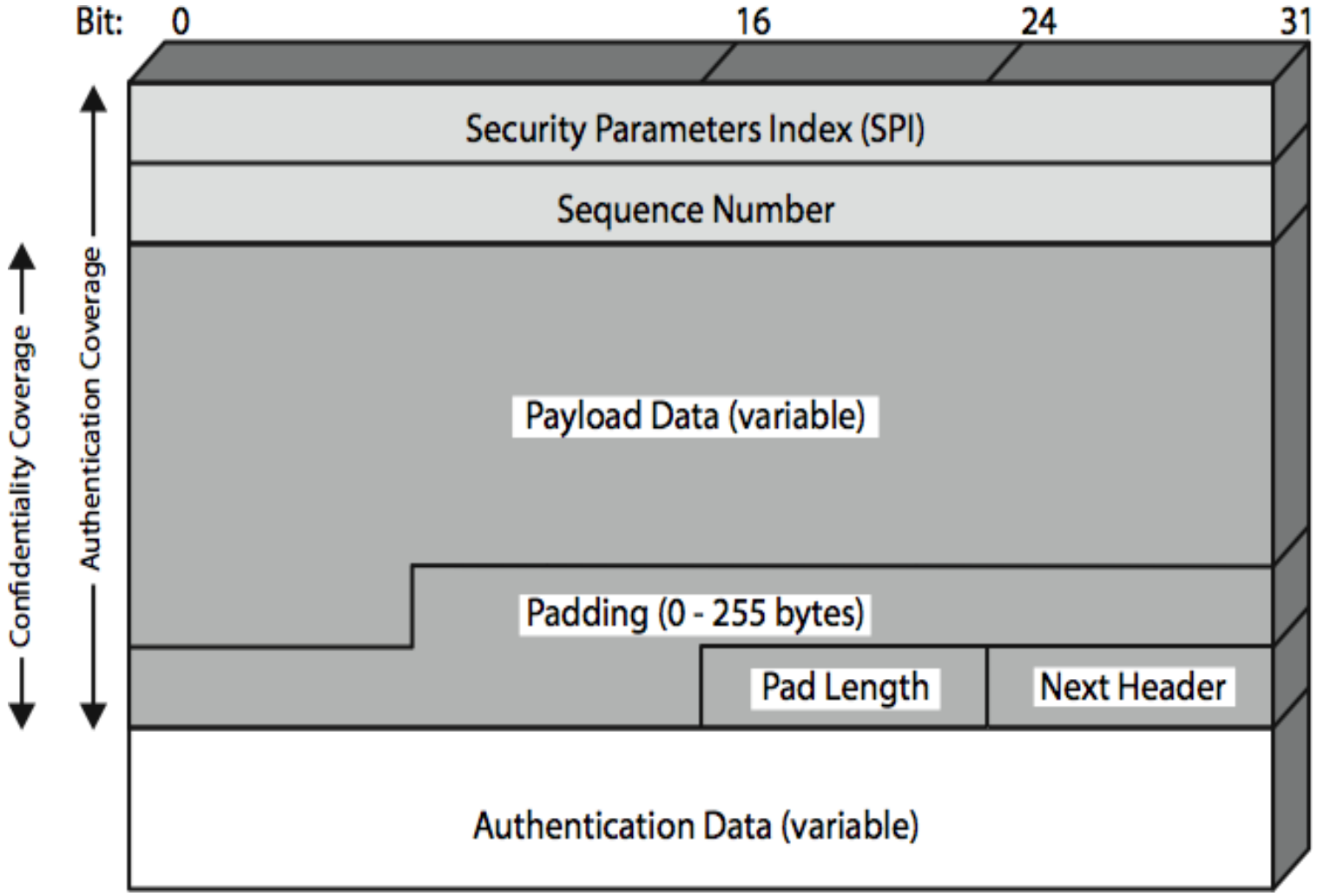
# Transport & Tunnel Modes



# Encapsulating Security Payload (ESP)

- Provides message content confidentiality & limited traffic flow confidentiality
- Can optionally provide the same authentication services as AH
- Supports range of ciphers, modes, padding
  - incl. DES, Triple-DES, RC5, IDEA, CAST etc
  - CBC & other modes
  - padding needed to fill blocksize, fields, for traffic flow

# Encapsulating Security Payload



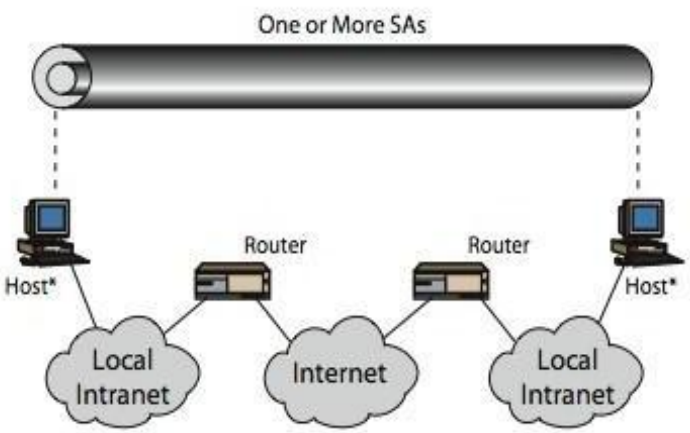
# Transport vs Tunnel Mode ESP

- Transport mode is used to encrypt & optionally authenticate IP data
  - data protected but header left in clear
  - can do traffic analysis but is efficient
  - good for ESP host to host traffic
- Tunnel mode encrypts entire IP packet
  - add new header for next hop
  - good for VPNs, gateway to gateway security

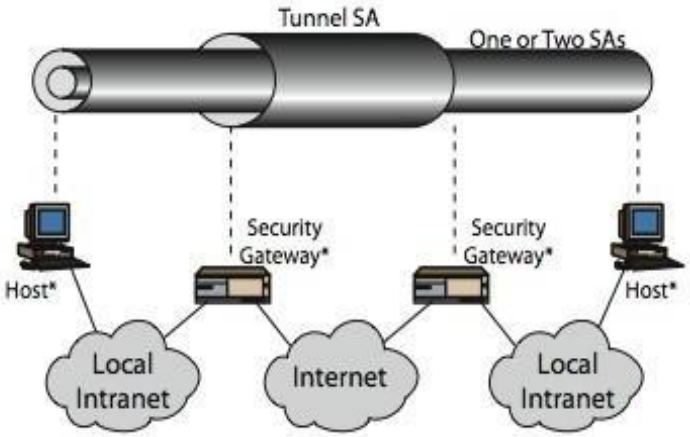
# Combining Security Associations

- SA's can implement either AH or ESP
- To implement both need to combine SA's
  - form a security association bundle
  - may terminate at different or same endpoints
  - combined by
    - transport adjacency
    - iterated tunneling
- Issue of authentication & encryption order

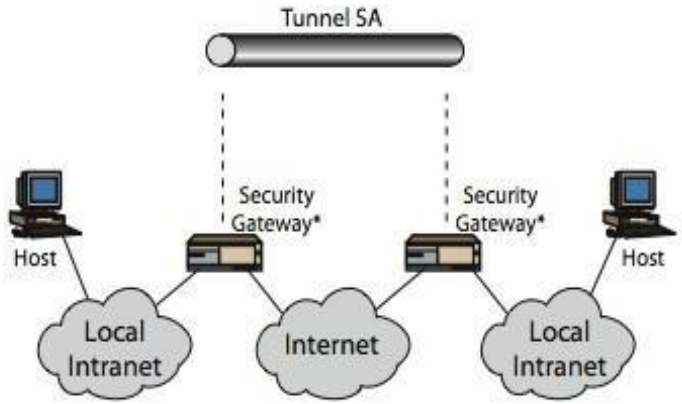
# Combining Security Associations



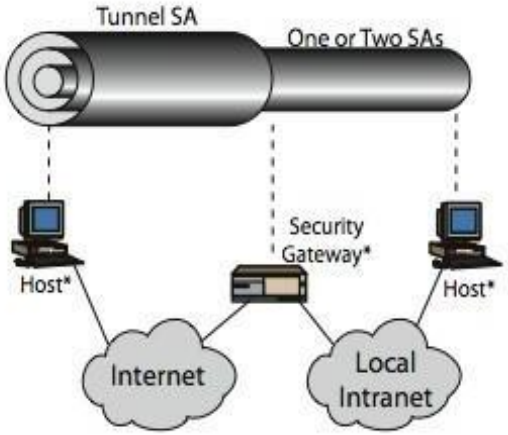

(a) Case 1




(c) Case 3




(b) Case 2




(d) Case 4

# Key Management

- Handles key generation & distribution
- Typically need 2 pairs of keys
  - 2 per direction for AH & ESP
- Manual key management
  - sysadmin manually configures every system
- Automated key management
  - automated system for on demand creation of keys for SA's in large systems
  - has Oakley & ISAKMP elements

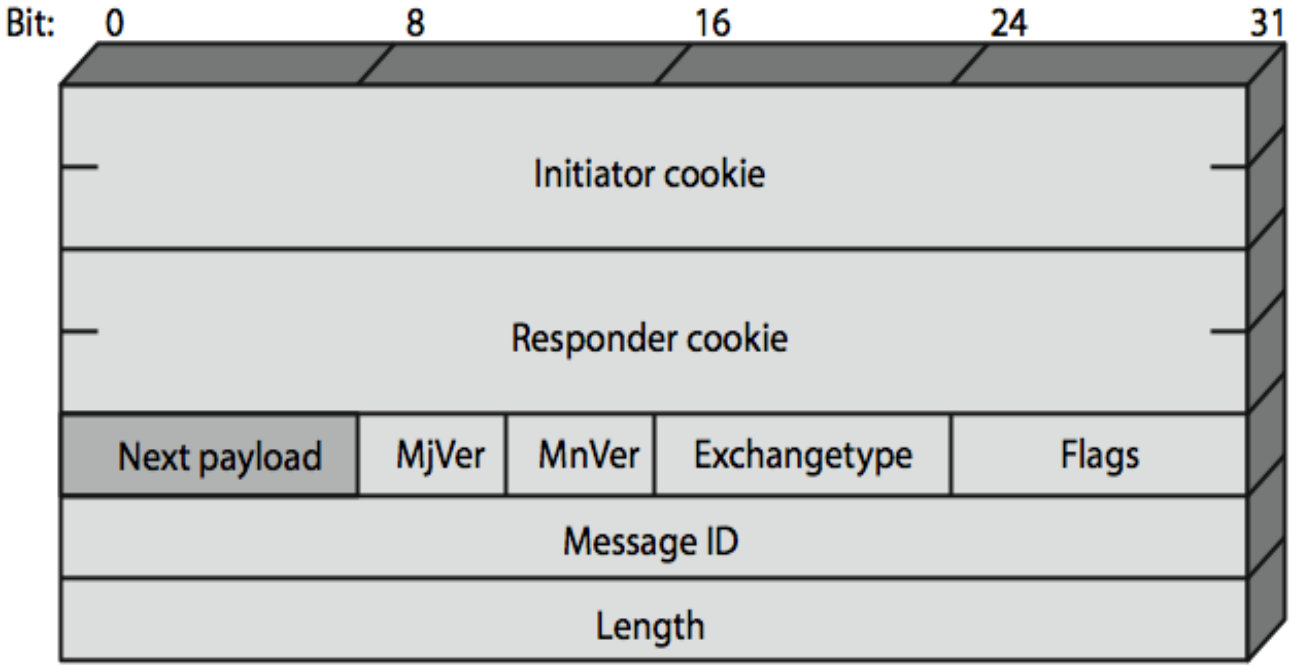
# Oakley

- A key exchange protocol
- Based on Diffie-Hellman key exchange
- Adds features to address weaknesses
  - cookies, groups (global params), nonces, DH key exchange with authentication
- Can use arithmetic in prime fields or elliptic curve fields

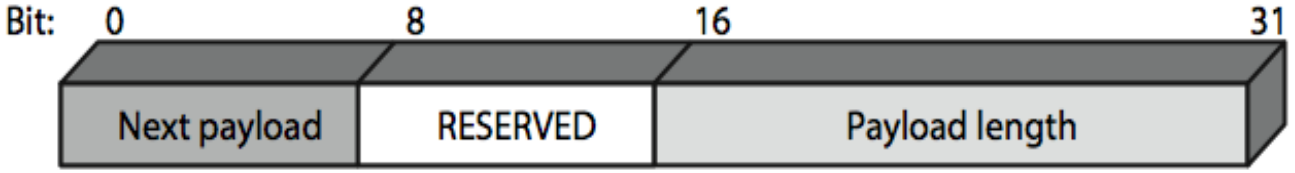
# ISAKMP

- Internet Security Association and Key Management Protocol
- Provides framework for key management
- Defines procedures and packet formats to establish,  negotiate, modify, & delete SAs
- Independent of key exchange protocol, encryption alg, & authentication method

# ISAKMP




(a) ISAKMP Header




(b) Generic Payload Header

# ISAKMP Payloads & Exchanges

- Have a number of ISAKMP payload types:
  - Security, Proposal, Transform, Key, Identification, Certificate, Certificate, Hash, Signature, Nonce, Notification, Delete
- ISAKMP has framework for 5 types of message exchanges:
  - base, identity protection, authentication only, aggressive,
    - informational


# UNIT– V
# WEB SECURITY

| CLOs | Course Learning Outcome |
| --- | --- |
| CLO1 | Describe the security socket layer and transport layer security for web security. |
| CLO2 | Demonstrate various types of intrusion detection techniques. |
| CLO3 | Understand various types of viruses and its vulnerabilities. |
| CLO4 | Describe various types of firewalls and analyze the security levels of these. |

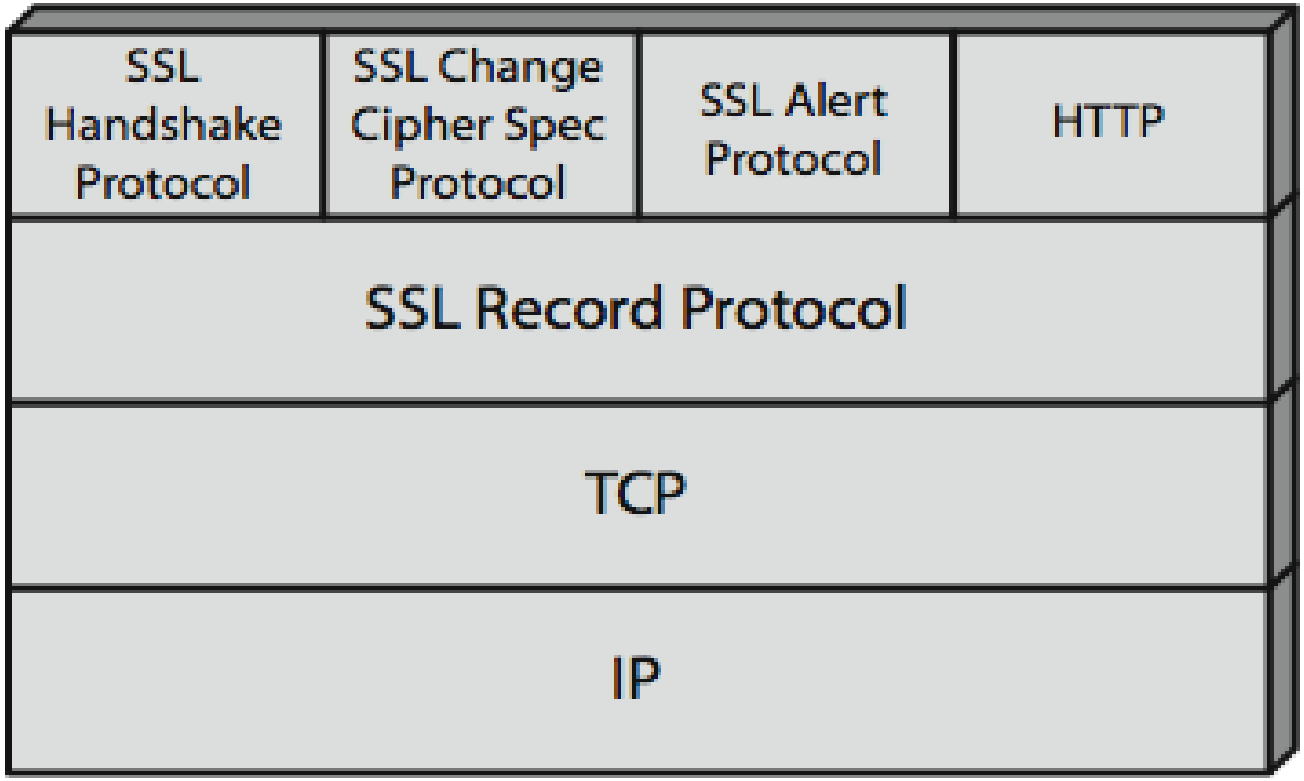
# Web Security

- Web now widely used by business, government, individuals
- but Internet & Web are vulnerable
- have a variety of threats
  - integrity
  - confidentiality
  - denial of service
  - authentication
- need added security mechanisms

# SSL (Secure Socket Layer)

- Transport layer security service
- Originally developed by Netscape
- Version 3 designed with public input
- Subsequently became Internet standard known as TLS (Transport Layer Security)
- Uses TCP to provide a reliable end-to-end service
- SSL has two layers of protocols

# SSL Architecture



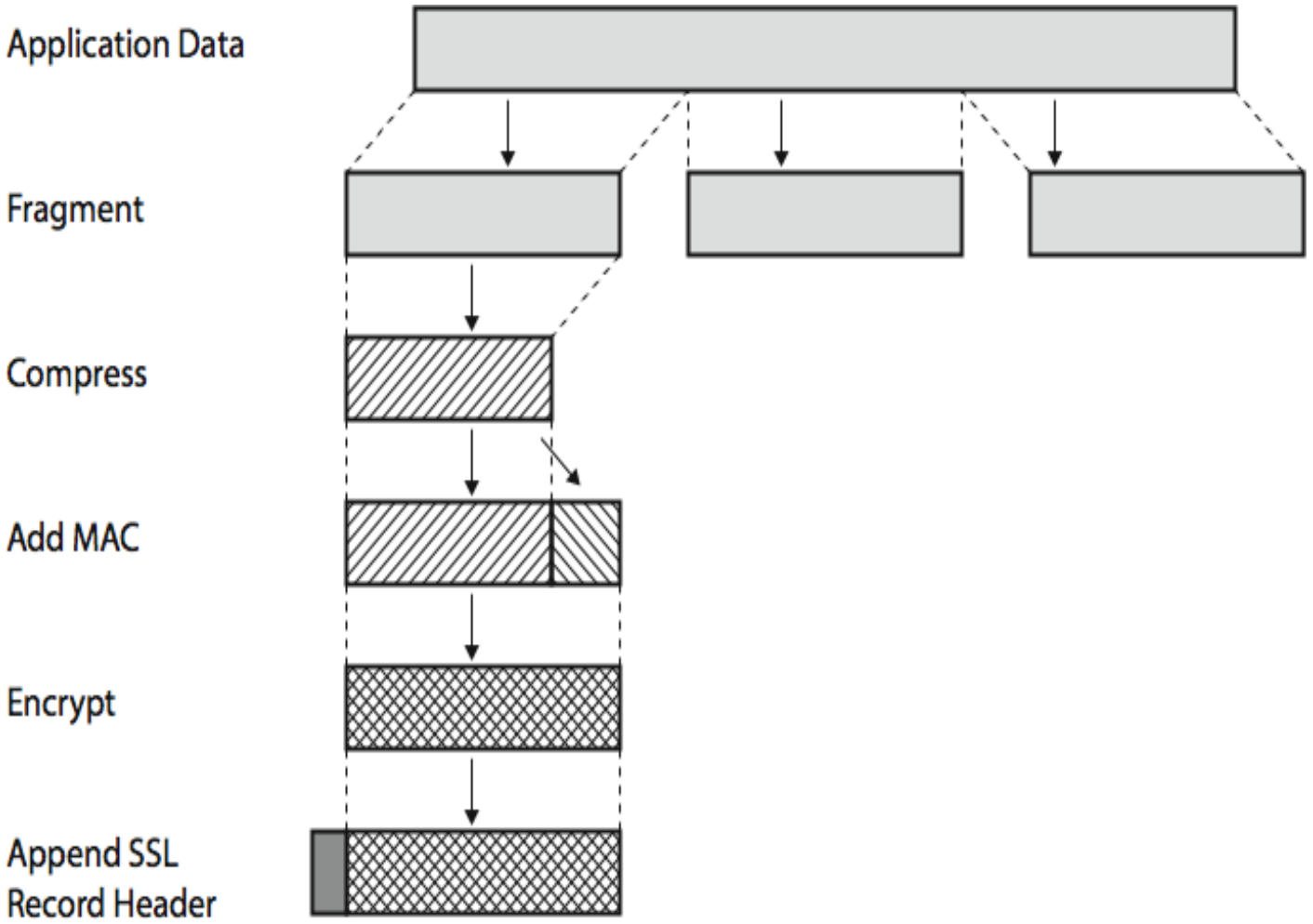
# SSL Architecture

- **SSL connection**
  - A transient, peer-to-peer, communications link
  - Associated with 1 SSL session
- **SSL session**
  - An association between client & server
  - Created by the Handshake Protocol
  - Define a set of cryptographic parameters
  - May be shared by multiple SSL connections

# SSL Record Protocol Services

- **Message integrity**
  - using a MAC with shared secret key
  - similar to HMAC but with different padding
- **Confidentiality**
  - using symmetric encryption with a shared secret key definedby Handshake Protocol
  - AES, IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
  - message is compressed before encryption

# SSL Record Protocol Operation



# SSL Change Cipher Spec Protocol

- One of 3 SSL specific protocols which use the SSL Record protocol
- A single message
- Causes pending state to become current
- Hence updating the cipher suite in use

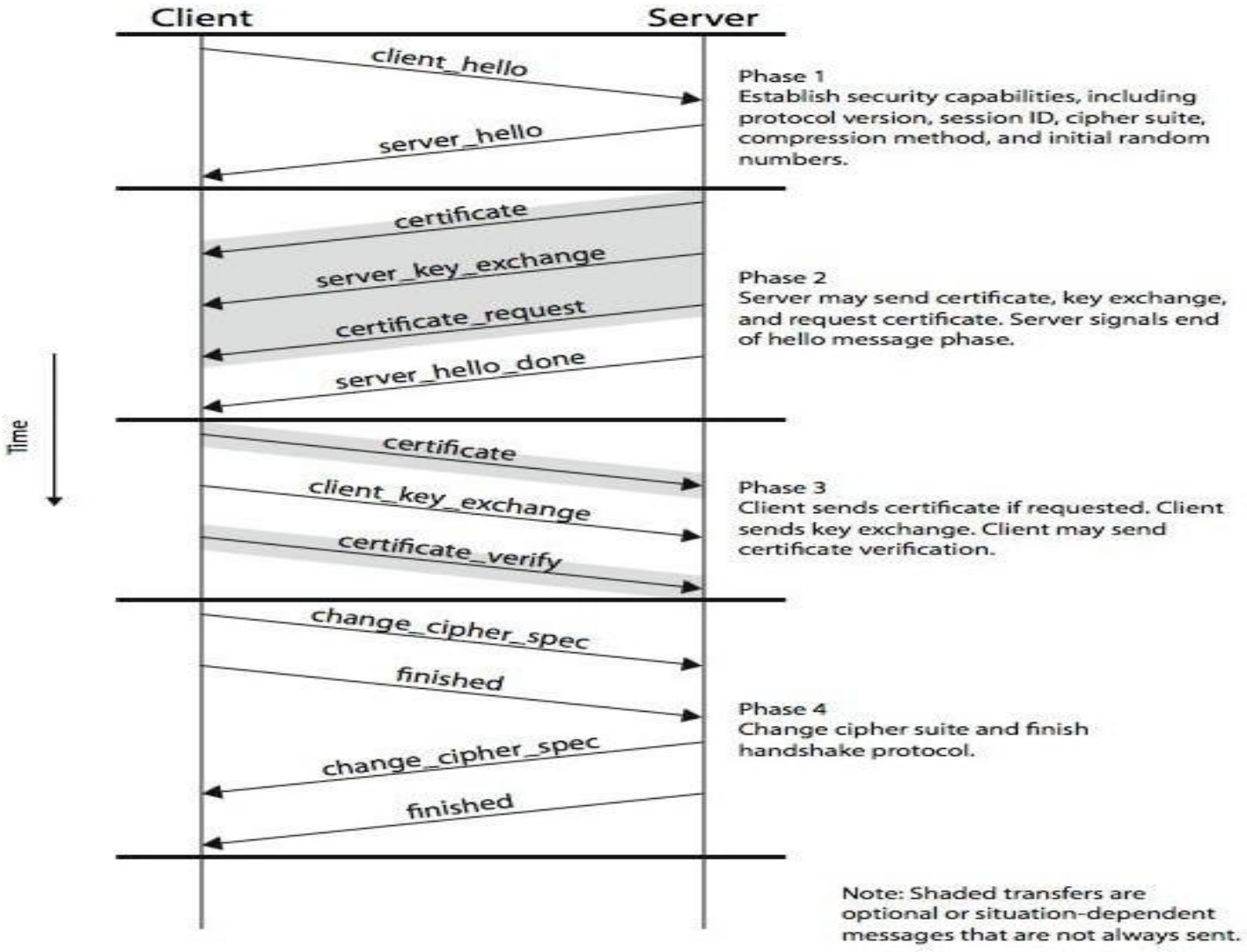
# SSL Alert Protocol

- Conveys SSL-related alerts to peer entity
- Severity warning or fatal
- Specific alert
  - fatal: unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
  - warning: close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- Compressed & encrypted like all SSL data

# SSL Handshake Protocol

- Allows server & client to:
  - authenticate each other
  - to negotiate encryption & MAC algorithms
  - to negotiate cryptographic keys to be used
- Comprises a series of messages in phases
  - Establish Security Capabilities
  - Server Authentication and Key Exchange
  - Client Authentication and Key Exchange
  - Finish

# SSL Handshake Protocol



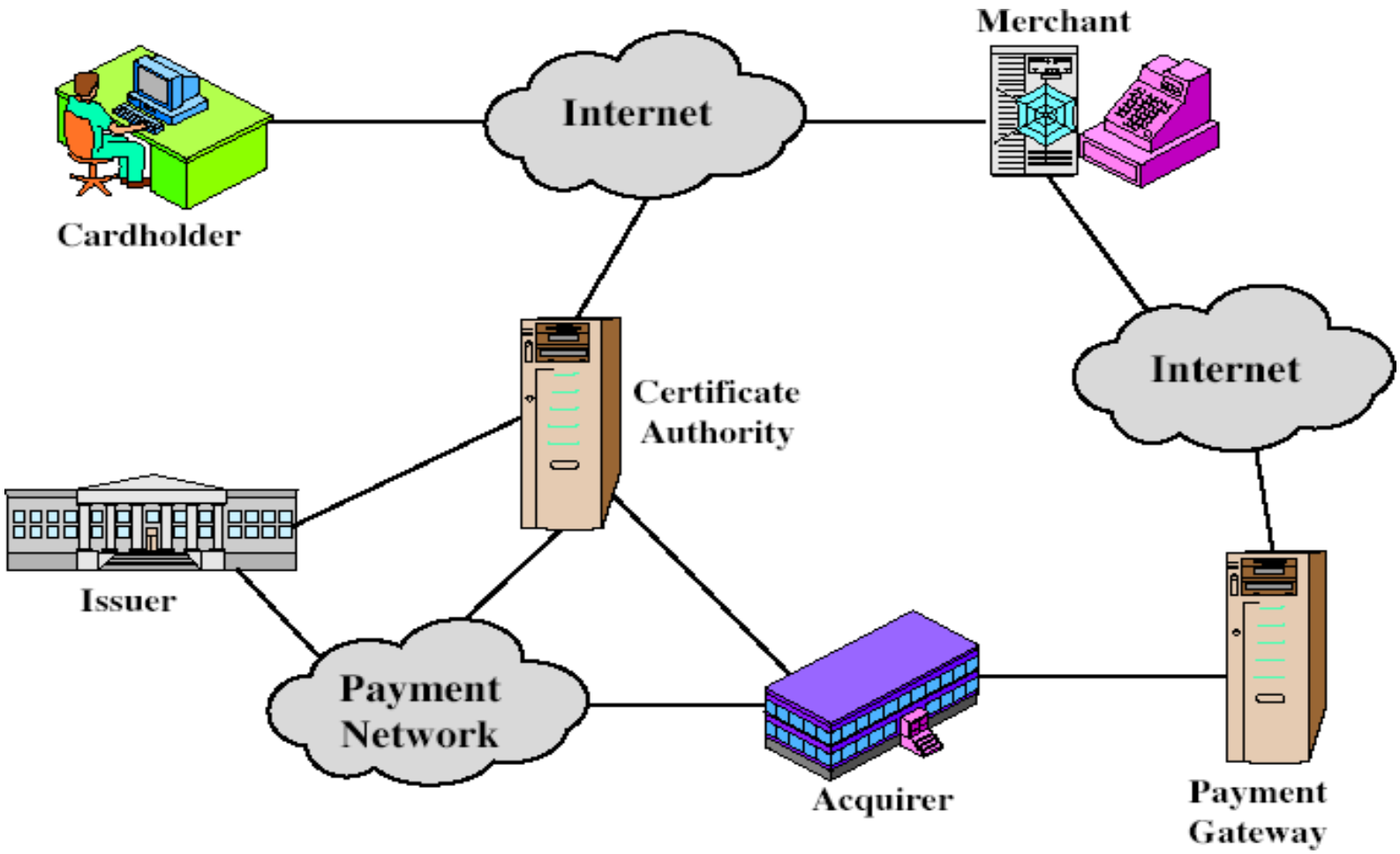
# TLS (Transport Layer Security)

- IETF standard RFC 2246 similar to SSLv3
- with minor differences
  - in record format version number
  - uses HMAC for MAC
  - a pseudo-random function expands secrets
  - has additional alert codes
  - some changes in supported ciphers
  - changes in certificate types & negotiations
  - changes in crypto computations & padding

# Secure Electronic Transactions (SET)

- Open encryption & security specification
- To protect Internet credit card transactions
- Developed in 1996 by Mastercard, Visa etc
- Not a payment system
- Rather a set of security protocols & formats
  - secure communications amongst parties
  - trust from use of X.509v3 certificates
  - privacy by restricted info to those who need it

# SET Components



# SET Transaction

- customer opens account
- customer receives a certificate
- merchants have their own certificates
- customer places an order
- merchant is verified
- order and payment are sent
- merchant requests payment authorization
- merchant confirms order
- merchant provides goods or service
- merchant requests payment

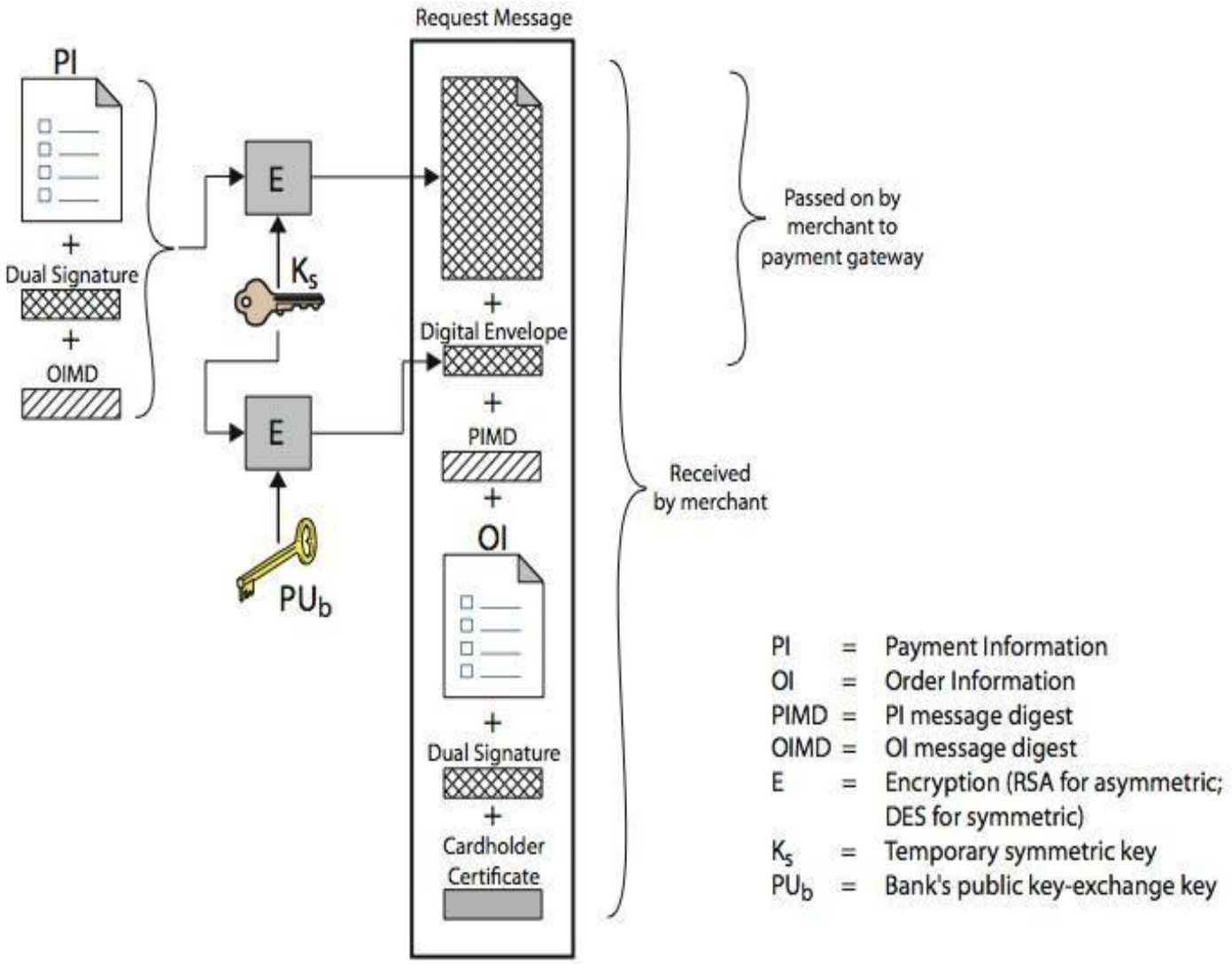
# Dual Signature

- Customer creates dual messages
  - order information (OI) for merchant
  - payment information (PI) for bank
- Neither party needs details of other
- But **must** know they are linked
- Use a dual signature for this
  - signed concatenated hashes of OI & PI
    - $DS=E(PR_c, [H(H(PI)||H(OI))])$

# SET Purchase Request

- SET purchase request exchange consists of four messages
  - Initiate Request - get certificates
  - Initiate Response - signed response
  - Purchase Request - of OI & PI
  - Purchase Response - ack order

# Purchase Request – Customer

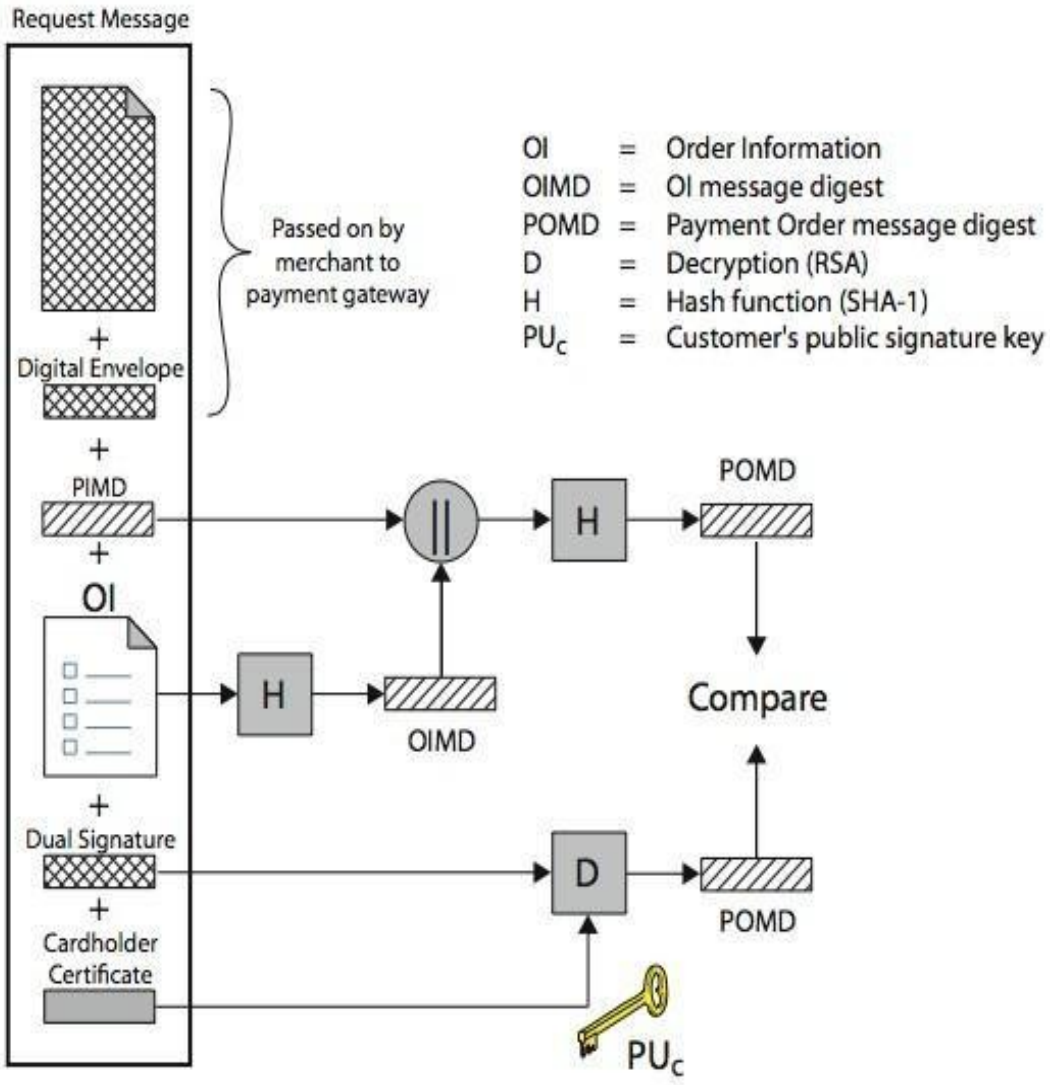



PI = Payment Information
OI = Order Information
PIMD = PI message digest
OIMD = OI message digest
E = Encryption (RSA for asymmetric; DES for symmetric)
$K_s$ = Temporary symmetric key
$PU_b$ = Bank's public key-exchange key

# Purchase Request – Merchant

- Verifies cardholder certificates using CA sigs
- Verifies dual signature using customer's public signature key to ensure order has not been tampered with in transit & that it was signed using cardholder's private signature key
- Processes order and forwards the payment information to the payment gateway for authorization (described later)
- Sends a purchase response to cardholder

# Purchase Request – Merchant



# Payment Gateway Authorization

- Verifies all certificates
- Decrypts digital envelope of authorization block to obtain symmetric key & then decrypts authorization block
- Verifies merchant's signature on authorization block decrypts digital envelope of payment block to obtain symmetric key & then decrypts payment block
- Verifies dual signature on payment block
- Verifies that transaction ID received from merchant matches that in PI received (indirectly) from customer

# Payment Capture

- Merchant sends payment gateway a payment capture request
- Gateway checks request
- Then causes funds to be transferred to merchants account
- Notifies merchant using capture response

# Intruders

- Clearly a growing publicized problem
  - from “Wily Hacker” in 1986/87
  - to clearly escalating CERT stats
- May seem benign, but still cost resources
- May use compromised system to launch other attacks
- Mwareness of intruders has led to the development of CERTs

# Intrusion Techniques

- Aim to gain access and/or increase privileges on a system
- Basic attack methodology
  - target acquisition and information gathering
  - initial access
  - privilege escalation
  - covering tracks
- Key goal often is to acquire passwords
- So then exercise access rights of owner

# Password Guessing

- One of the most common attacks
- Attacker knows a login (from email/web page etc)
- Then attempts to guess password for it
  - defaults, short passwords, common word searches
  - user info (variations on names, birthday, phone, common words/interests)
  - exhaustively searching all possible passwords
- Check by login or against stolen password file
- Success depends on password chosen by user
- Surveys show many users choose poorly

# Password Capture

- Another attack involves **password capture**
  - watching over shoulder as password is entered
  - using a trojan horse program to collect
  - monitoring an insecure network login
    - eg. telnet, FTP, web, email
  - extracting recorded info after successful login (web history/cache, last number dialed etc)
- Using valid login/password can impersonate user
- Users need to be educated to use suitable precautions/countermeasures

# Intrusion Detection

- Inevitably will have security failures
- So need also to detect intrusions so can
  - block if detected quickly
  - act as deterrent
  - collect info to improve security
- Assume intruder will behave differently to a legitimate user
  - but will have imperfect distinction between

# Approaches to Intrusion Detection

- Statistical anomaly detection
  - threshold
  - profile based
- Rule-based detection
  - anomaly
  - penetration identification

# Audit Records

- Fundamental tool for intrusion detection
- Native audit records
  - part of all common multi-user O/S
  - already present for use
  - may not have info wanted in desired form
- Detection-specific audit records
  - created specifically to collect wanted info
  - at cost of additional overhead on system

# Statistical Anomaly Detection

- Threshold detection
  - count occurrences of specific event over time
  - if exceed reasonable value assume intrusion
  - alone is a crude & ineffective detector
- Profile based
  - characterize past behavior of users
  - detect significant deviations from this
  - profile usually multi-parameter

# Audit Record Analysis

- Foundation of statistical approaches
- Analyze records to get metrics over time
  - counter, gauge, interval timer, resource use
- Use various tests on these to determine if current behavior is acceptable
  - mean & standard deviation, multivariate, markov process, time series, operational
- Key advantage is no prior knowledge used

# Rule-Based Intrusion Detection

- Observe events on system & apply rules to decide if activity is suspicious or not
- Rule-based anomaly detection
  - Analyze historical audit records to identify usage patterns & auto-generate rules for them
  - Then observe current behavior & match against rules to see if conforms
  - Like statistical anomaly detection does not require prior knowledge of security flaws

# Rule-Based Intrusion Detection

- Rule-based penetration identification
  - uses expert systems technology
  - with rules identifying known penetration, weakness
    - patterns, or suspicious behavior
  - compare audit records or states against rules
  - rules usually machine & O/S specific
  - rules are generated by experts who interview & codify knowledge of security admins
  - quality depends on how well this is done

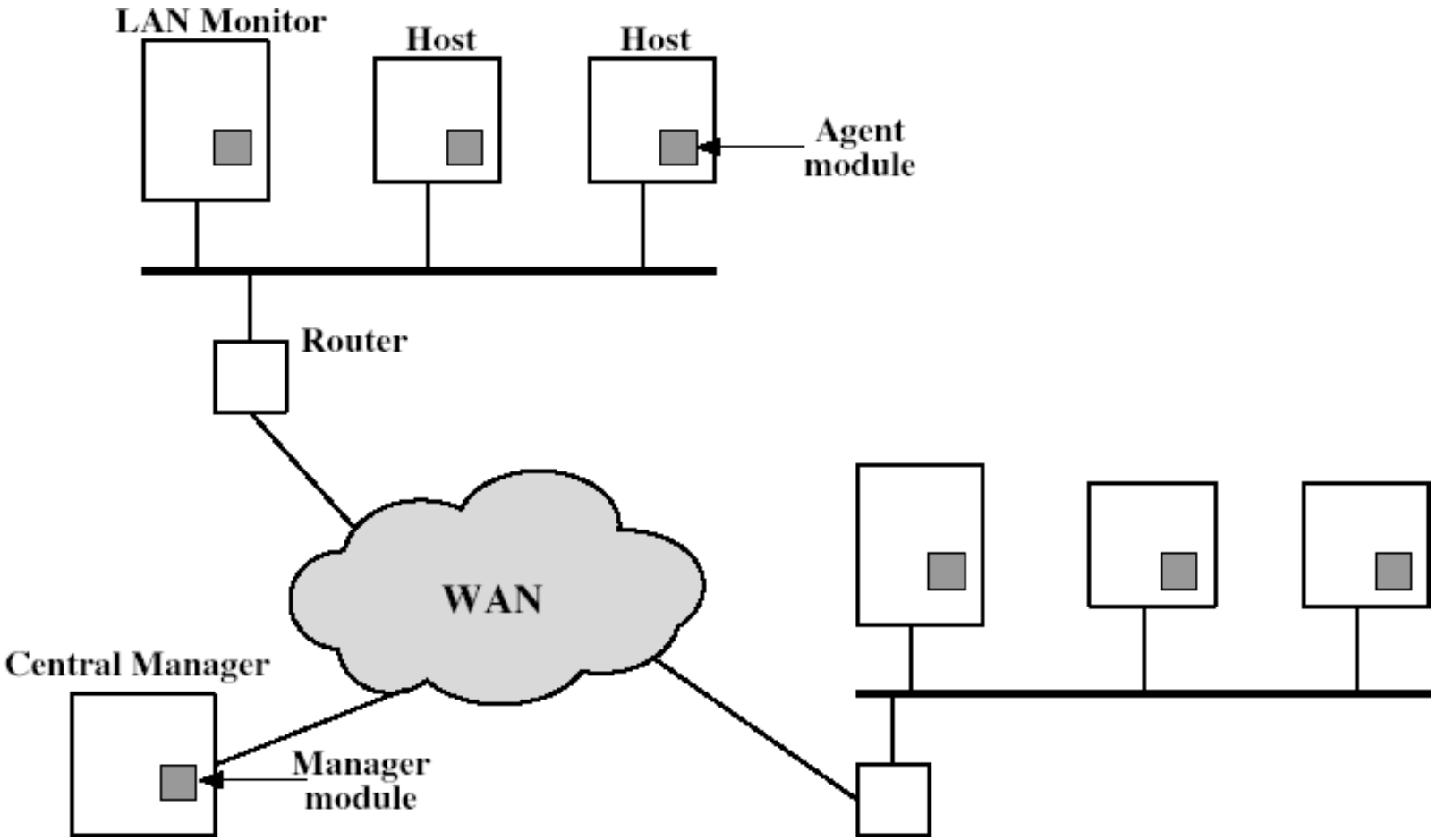
# Base-Rate Fallacy

- Practically an intrusion detection system needs to detect a substantial percentage of intrusions with few false alarms
  - if too few intrusions detected -> false security
  - if too many false alarms -> ignore / waste time
- This is very hard to do
- Existing systems seem not to have a good record

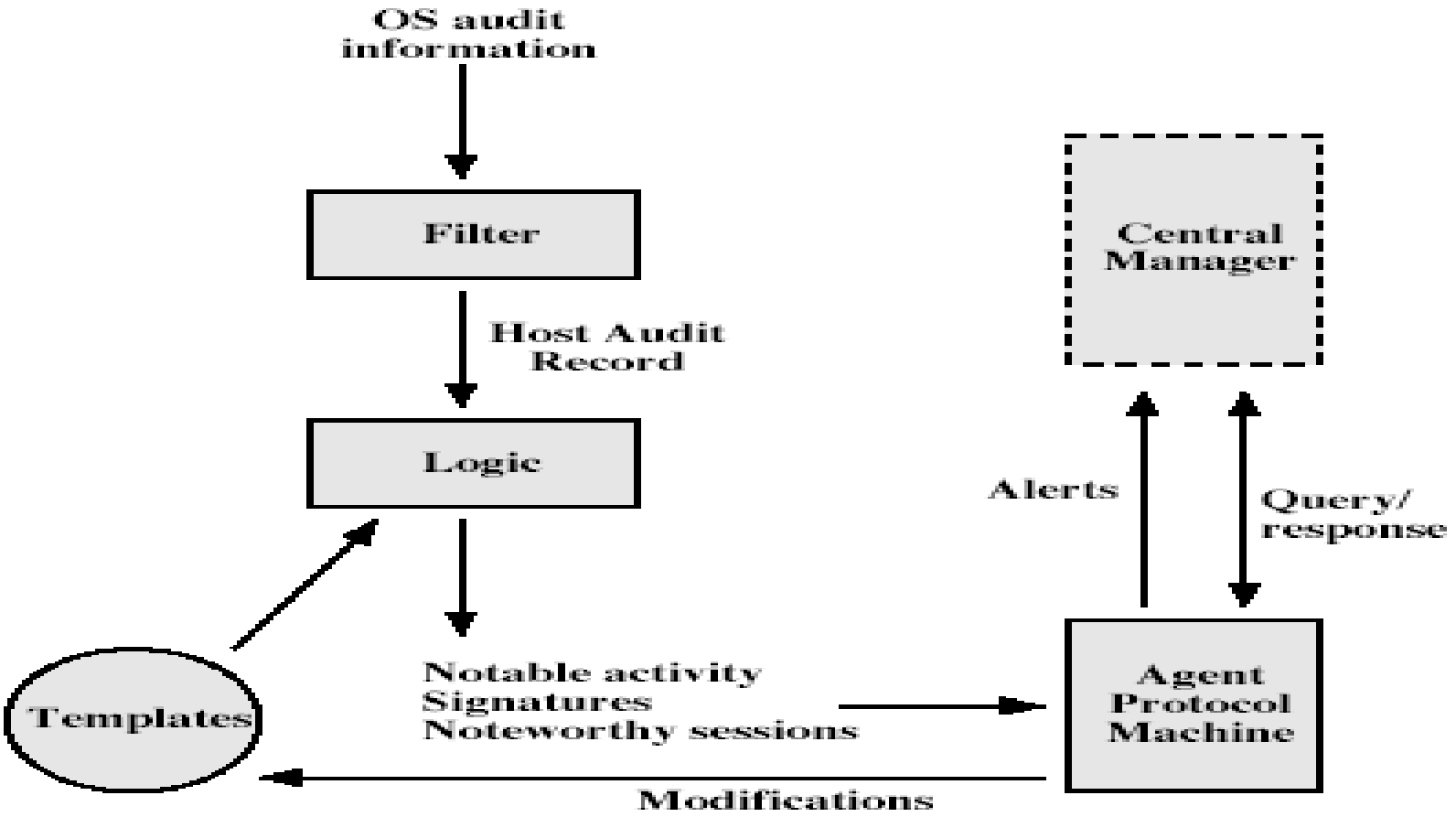
# Distributed Intrusion Detection

- Traditional focus is on single systems
- But typically have networked systems
- More effective defense has these working together to detect intrusions
- Issues
  - dealing with varying audit record formats
  - integrity & confidentiality of networked data
  - centralized or decentralized architecture

# Distributed Intrusion Detection - Architecture



# Distributed Intrusion Detection – Agent Implementation



# Honeypots

- Decoy systems to lure attackers
  - away from accessing critical systems
  - to collect information of their activities
  - to encourage attacker to stay on system so administrator can
    - respond
- Are filled with fabricated information
- Instrumented to collect detailed information on attackers activities
- Single or multiple networked systems
- cf IETF Intrusion Detection WG standards

# Password Management

- Front-line defense against intruders
- Users supply both:
  - login – determines privileges of that user
  - password – to identify them
- Passwords often stored encrypted
  - Unix uses multiple DES (variant with salt)
  - more recent systems use crypto hash function
- Should protect password file on system

# Password Studies

- Purdue 1992 - many short passwords
- Klein 1990 - many guessable passwords
- Conclusion is that users choose poor passwords too often
- Need some approach to counter this

# Managing Passwords - Education

- Can use policies and good user education
- Educate on importance of good passwords
- Give guidelines for good passwords
  - minimum length (>6)
  - require a mix of upper & lower case letters, numbers,punctuation
  - not dictionary words
- But likely to be ignored by many users

# Managing Passwords - Computer Generated

- Let computer create passwords
- If random likely not memorisable, so will be written down (sticky label syndrome)
- Even pronounceable not remembered
- Have history of poor user acceptance
- FIPS PUB 181 one of best generators
  - has both description & sample code
  - generates words from concatenating random pronounceable syllables

# Managing Passwords - Reactive Checking

- Reactively run password guessing tools
- Note that good dictionaries exist for almost any language/interest group
- Cracked passwords are disabled
- But is resource intensive
- Bad passwords are vulnerable till found

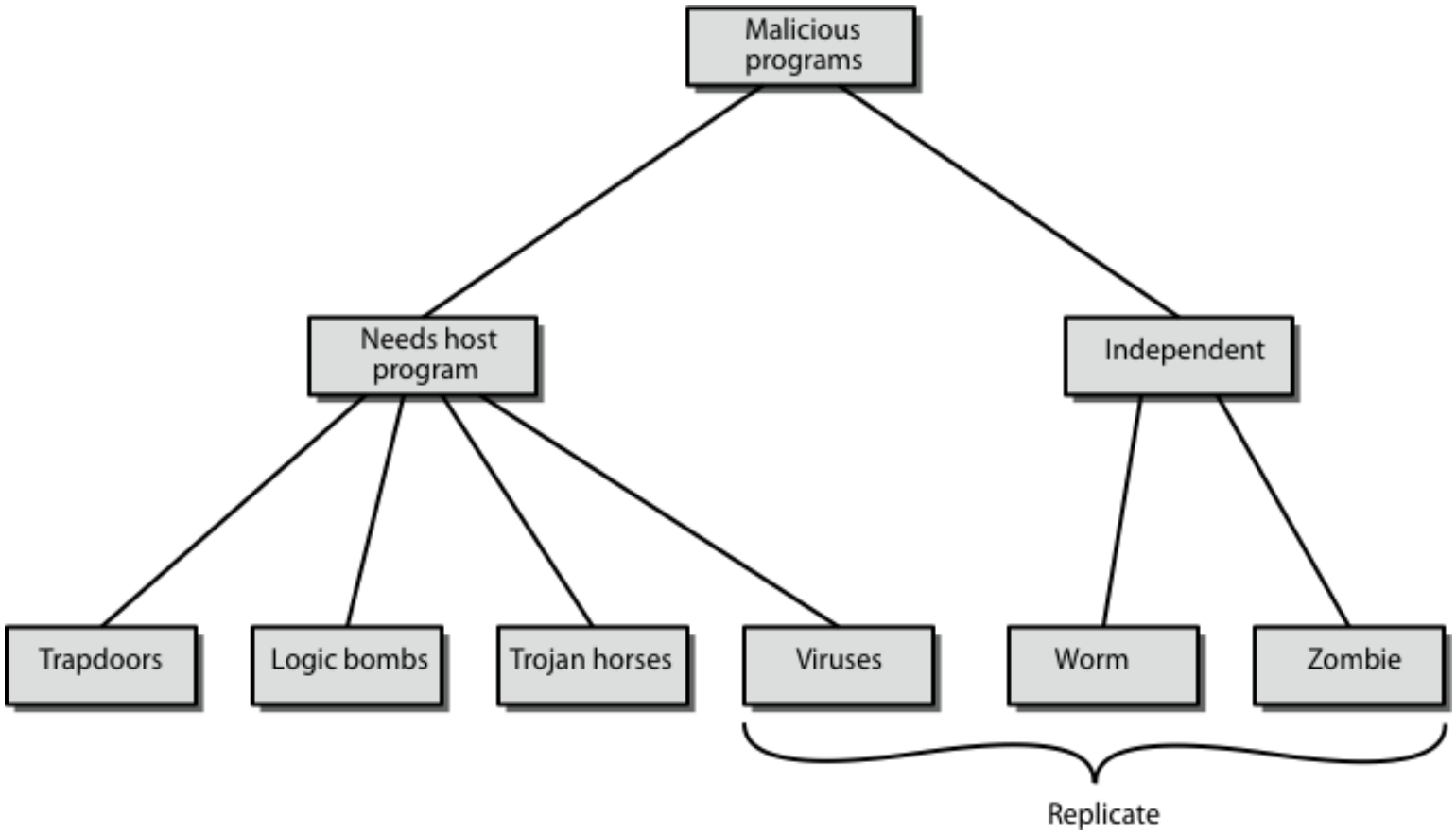
# Managing Passwords - Proactive Checking

- Most promising approach to improving password security
- Allow users to select own password
- But have system verify it is acceptable
  - simple rule enforcement (see earlier slide)
  - compare against dictionary of bad passwords
  - use algorithmic (markov model or bloom filter) to detect poor choices

# Viruses and Other Malicious Content

- Computer viruses have got a lot of publicity
- One of a family of **malicious software**
- Effects usually obvious
- Have figured in news reports, fiction, movies (often exaggerated)
- Getting more attention than deserve
- Are a concern though

# Malicious Software



# Backdoor or Trapdoor

- Secret entry point into a program allows those who know access bypassing usual security procedures
- Have been commonly used by developers
- A threat when left in production programs allowing exploited by attackers
- Very hard to block in O/S
- Requires good s/w development & update

# Logic Bomb

- One of oldest types of malicious software
- Code embedded in legitimate program
- Activated when specified conditions met
  - eg presence/absence of some file
  - particular date/time
  - particular user
- When triggered typically damage system
  - modify/delete files/disks, halt machine, etc

# Trojan Horse

- Program with hidden side-effects
- Which is usually superficially attractive
  - eg game, s/w upgrade etc
- When run performs some additional tasks
  - allows attacker to indirectly gain access they do not have directly
- Often used to propagate a virus/worm or install a backdoor
- or simply to destroy data

# Zombie

- Program which secretly takes over another networked computer
- Then uses it to indirectly launch attacks
- Often used to launch distributed denial of service (DDoS) attacks
- Exploits known flaws in network systems

# Viruses

- A piece of self-replicating code attached to some other code
  - cf biological virus
- Both propagates itself & carries a payload
  - carries code to make copies of itself
  - as well as code to perform some covert task

# Virus Operation

- Virus phases:
  - Dormant – waiting on trigger event
  - Propagation – replicating to programs/disks
  - Triggering – by event to execute payload
  - Execution – of payload
- Details usually machine/OS specific
  - Exploiting features/weaknesses

# Virus Structure

```
program V :=
   {goto
   main;
   1234567;
   subroutine infect-executable := {loop:
                    file := get-random-executable-file;
                    if (first-line-of-file = 1234567) then goto loop
                    else prepend V to file; }
   subroutine do-damage := {whatever damage is to be done}
   subroutine trigger-pulled := {return true if condition holds}
   main: main-program := {infect-executable;
                                    if trigger-pulled then do-damage;
                                    goto next;}
   next:
}
```

# Types of Viruses

- Can classify on basis of how they attack
- Parasitic virus
- Memory-resident virus
- Boot sector virus
- Stealth
- Polymorphic virus
- Metamorphic virus

# Macro Virus

- **Macro code** attached to some **data file**
- Interpreted by program using file
  - eg Word/Excel macros
  - esp. using auto command & command macros
- Code is now platform independent
- Is a major source of new viral infections
- Blur distinction between data and program files
- Classic trade-off: "ease of use" vs "security”
- Have improving security in Word etc
- Are no longer dominant virus threat

# Email Virus

- Spread using email with attachment containing a macro virus
  - cf Melissa
- Triggered when user opens attachment
- or worse even when mail viewed by using scripting features in mail agent
- Hence propagate very quickly
- Usually targeted at Microsoft Outlook mail agent & Word/Excel documents
- Need better O/S & application security

# Worms

- Replicating but not infecting program
- Typically spreads over a network
  - cf Morris Internet Worm in 1988
  - led to creation of CERTs
- Using users distributed privileges or by exploiting system vulnerabilities
- Widely used by hackers to create zombie PC's, subsequently used for further attacks, esp DoS
- Major issue is lack of security of permanently connected systems, esp PC's

# Worm Operation

- Worm phases like those of viruses:
  - dormant
  - propagation
    - search for other systems to infect
    - establish connection to target remote system
    - replicate self onto remote system
  - triggering
  - execution

# Morris Worm

- Best known classic worm
- Released by Robert Morris in 1988
- Targeted Unix systems
- Using several propagation techniques
  - simple password cracking of local pw file
  - exploit bug in finger daemon
  - exploit debug trapdoor in sendmail daemon
- If any attack succeeds then replicated self

# Recent Worm Attacks

- New spate of attacks from mid-2001
- Code Red - used MS IIS bug
  - probes random IPs for systems running IIS
  - had trigger time for denial-of-service attack
  - 2nd wave infected 360000 servers in 14 hours
- Code Red 2 - installed backdoor
- Nimda - multiple infection mechanisms
- SQL Slammer - attacked MS SQL server
- Sobig.f - attacked open proxy servers
- Mydoom - mass email worm + backdoor

# Worm Techology

- Multiplatform
- Multiexploit
- Ultrafast spreading
- Polymorphic
- Metamorphic
- Transport vehicles
- Zero-day exploit

# Virus Countermeasures

- Best countermeasure is prevention
- But in general not possible
- Hence need to do one or more of:
  - detection - of viruses in infected system
  - identification - of specific infecting virus
  - removeal - restoring system to clean state

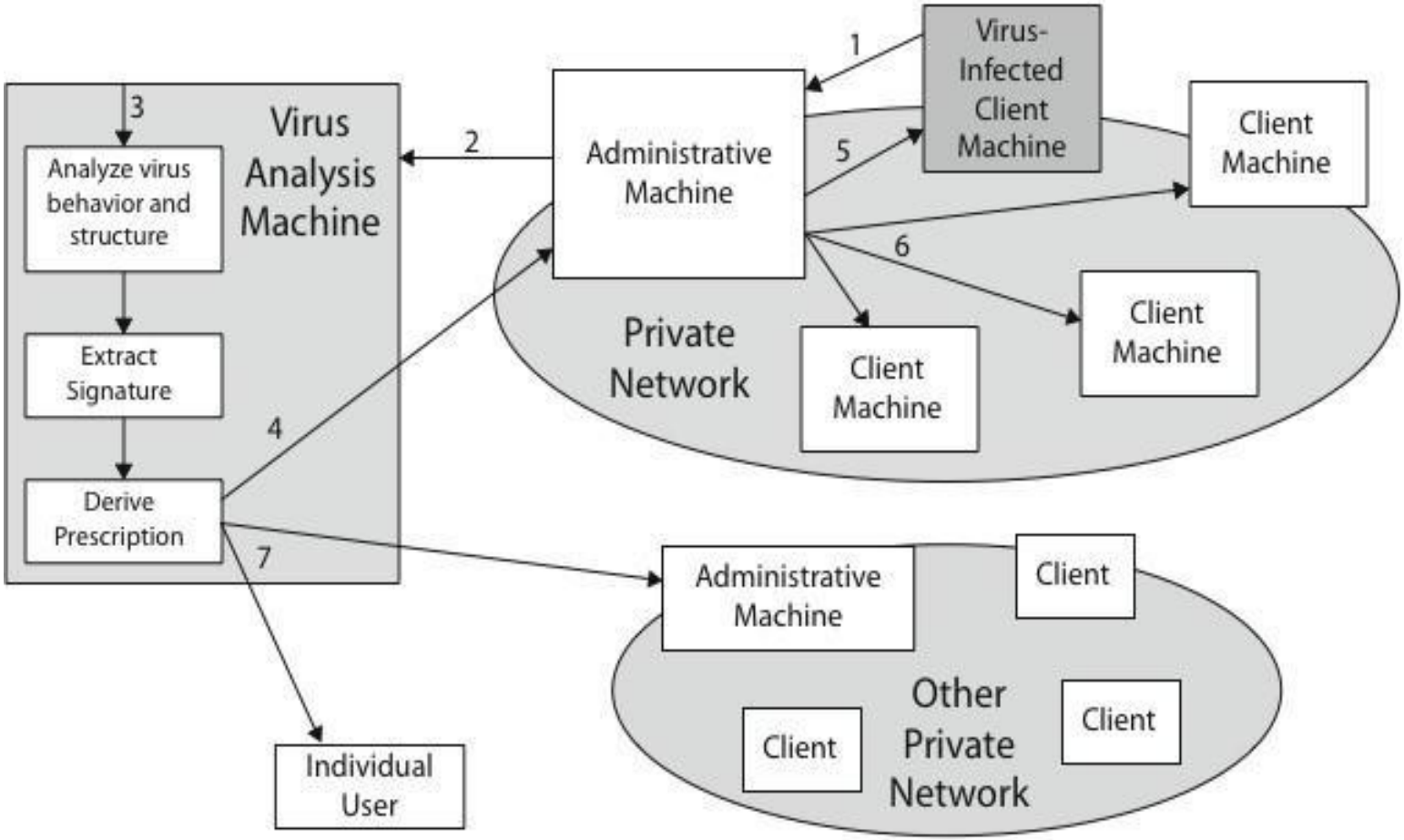
# Anti-Virus Software

- **first-generation**
  - scanner uses virus signature to identify virus
  - or change in length of programs
- **second-generation**
  - uses heuristic rules to spot viral infection
  - or uses crypto hash of program to spot changes
- **third-generation**
  - memory-resident programs identify virus by actions
- **fourth-generation**
  - packages with a variety of antivirus techniques
  - eg scanning & activity traps, access-controls
- arms race continues

# Advanced Anti-Virus Techniques

- Generic decryption
  - use CPU simulator to check program signature & behavior before actually running it
- Digital immune system (IBM)
  - general purpose emulation & virus detection
  - any virus entering org is captured, analyzed, detection/shielding created for it, removed

# Digital Immune System



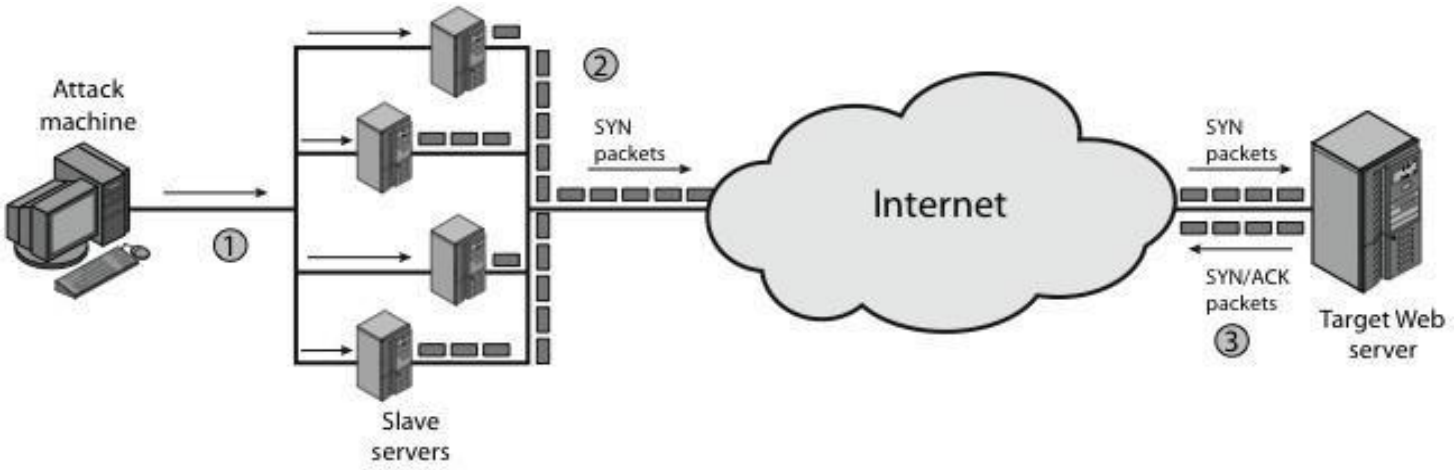
# Behavior-Blocking Software

- Integrated with host O/S monitors program behavior in real-time
  - eg file access, disk format, executable mods, system settings changes, network access
- For possibly malicious actions
  - if detected can block, terminate, or seek ok
- Has advantage over scanners
- But malicious code runs before detection

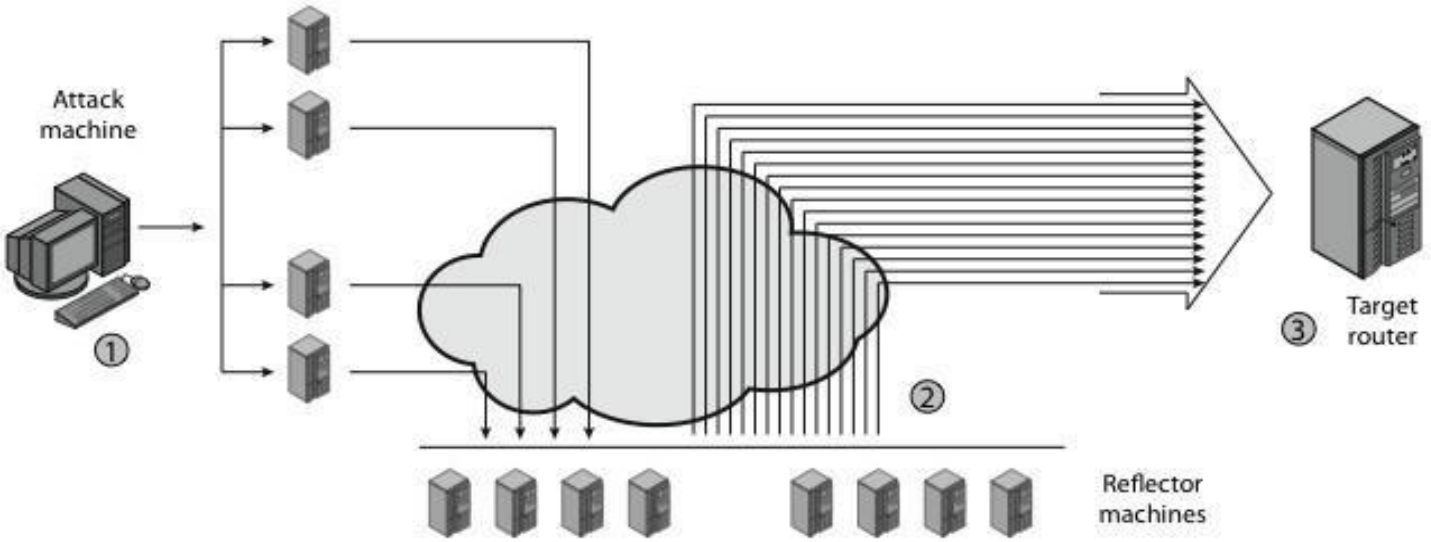
# Distributed Denial of Service Attacks (DDoS)

- Distributed Denial of Service (DDoS) attacks form a significant security threat
- making networked systems unavailable
- by flooding with useless traffic
- using large numbers of “zombies”
- growing sophistication of attacks
- defense technologies struggling to cope

# Distributed Denial of Service Attacks (DDoS)




(a) Distributed SYN flood attack




(a) Distributed ICMP attack

# Contructing the DDoS Attack Network

- Must infect large number of zombies
- Needs:
- Software to implement the DDoS attack
- An unpatched vulnerability on many systems
- Scanning strategy to find vulnerable systems
  - random, hit-list, topological, local subnet

# DDoS Countermeasures

- Three broad lines of defense:
  - attack prevention & preemption (before)
  - attack detection & filtering (during)
  - attack source traceback & ident (after)
- Huge range of attack possibilities
- Hence evolving countermeasures

# What is a Firewall?

- A choke point of control and monitoring
- Interconnects networks with differing trust
- Imposes restrictions on network services
  - only authorized traffic is allowed
- Auditing and controlling access
  - can implement alarms for abnormal behavior
- Provide NAT & usage monitoring
- Implement VPNs using IPSec
- Must be immune to penetration

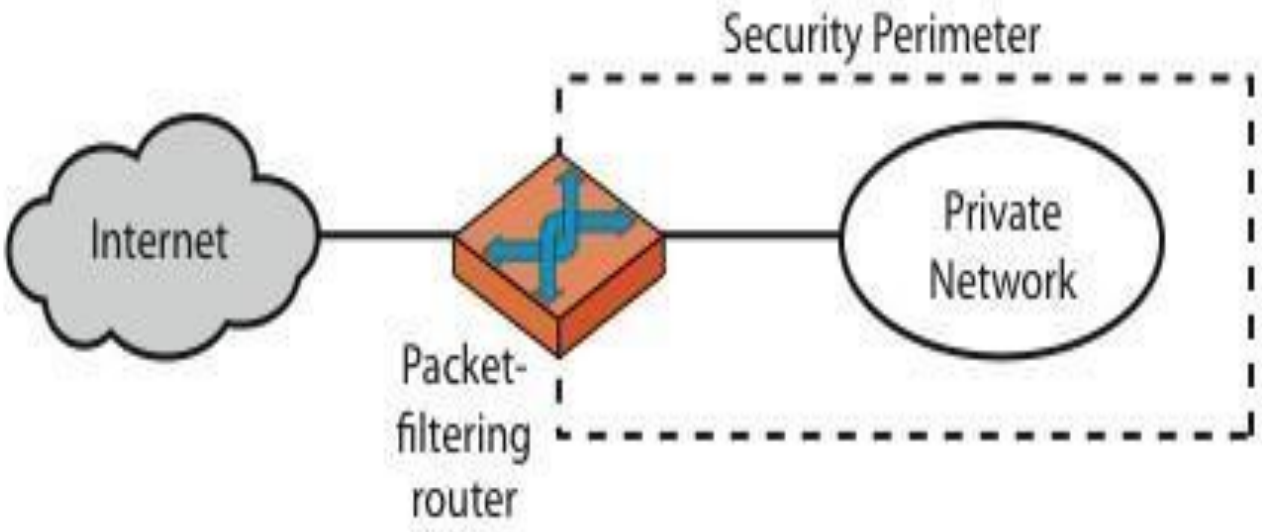
# Firewall Limitations

- Cannot protect from attacks bypassing it
  - eg sneaker net, utility modems, trusted organisations, trusted
    - services (eg SSL/SSH)
- Cannot protect against internal threats
  - eg disgruntled or colluding employees
- Cannot protect against transfer of all virus infected programs or files
  - because of huge range of O/S & file types

# Firewalls – Packet Filters

- Simplest, fastest firewall component
- Foundation of any firewall system
- Examine each IP packet (no context) and permit or deny according to rules
- Hence restrict access to services (ports)
- Possible default policies
  - that not expressly permitted is prohibited
  - that not expressly prohibited is permitted

# Firewalls – Packet Filters




(a) Packet-filtering router

# Firewalls – Packet Filters

Table 20.1 Packet-Filtering Examples

A

| action | ourhost | port | theirhost | port | comment |
|---|---|---|---|---|---|
| block | * | * | SPIGOT | * | we don't trust these people |
| allow | OUR-GW | 25 | * | * | connection to our SMTP port |

B

| action | ourhost | port | theirhost | port | comment |
|---|---|---|---|---|---|
| block | * | * | * | * | default |

C

| action | ourhost | port | theirhost | port | comment |
|---|---|---|---|---|---|
| allow | * | * | * | 25 | connection to their SMTP port |

D

| action | src | port | dest | port | flags | comment |
|---|---|---|---|---|---|---|
| allow | {our hosts} | * | * | 25 | | our packets to their SMTP port |
| allow | * | 25 | * | * | ACK | their replies |

E

| action | src | port | dest | port | flags | comment |
|---|---|---|---|---|---|---|
| allow | {our hosts} | * | * | * | | our outgoing calls |
| allow | * | * | * | * | ACK | replies to our calls |
| allow | * | * | * | >1024 | | traffic to nonservers |

# Attacks on Packet Filters

- IP address spoofing
  - fake source address to be trusted
  - add filters on router to block
- Source routing attacks
  - attacker sets a route other than default
  - block source routed packets
- Tiny fragment attacks
  - split header info over several tiny packets
  - either discard or reassemble before check

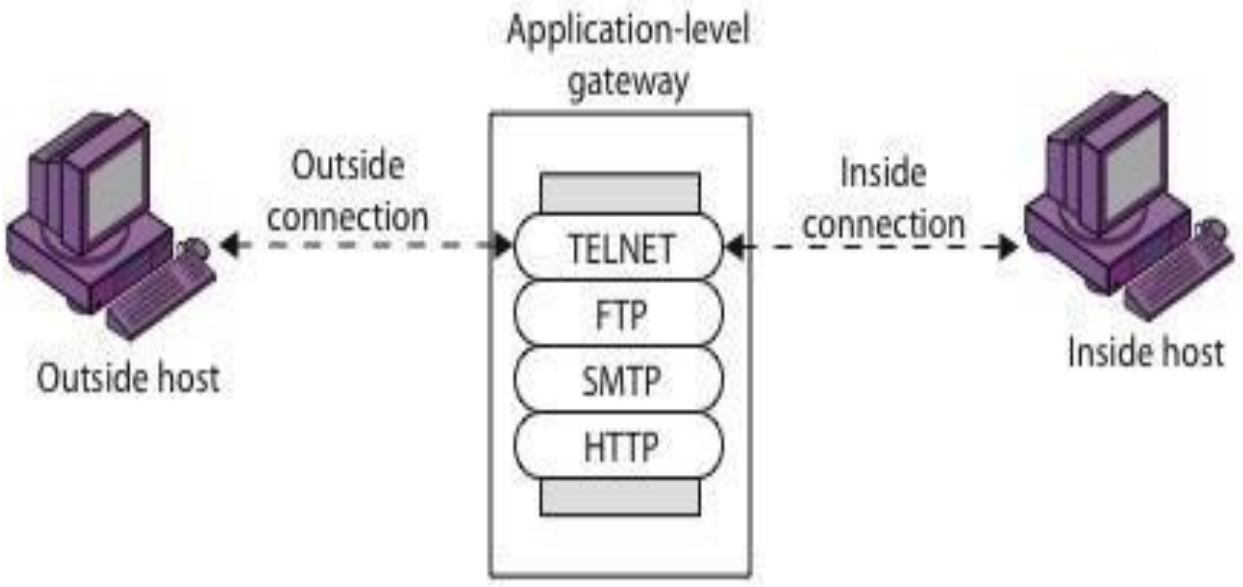
# Firewalls – Stateful Packet Filters

- Traditional packet filters do not examine higher layer context
  - ie matching return packets with outgoing flow
- Stateful packet filters address this need
- they examine each IP packet in context
  - keep track of client-server sessions
  - check each packet validly belongs to one
- Hence are better able to detect bogus packets out of context

# Firewalls - Application Level Gateway (Proxy)

- Have application specific gateway / proxy
- Has full access to protocol
  - user requests service from proxy
  - proxy validates request as legal
  - then actions request and returns result to user
  - can log / audit traffic at application level
- Need separate proxies for each service
  - some services naturally support proxying
  - others are more problematic

# Firewalls - Application Level Gateway (or Proxy)

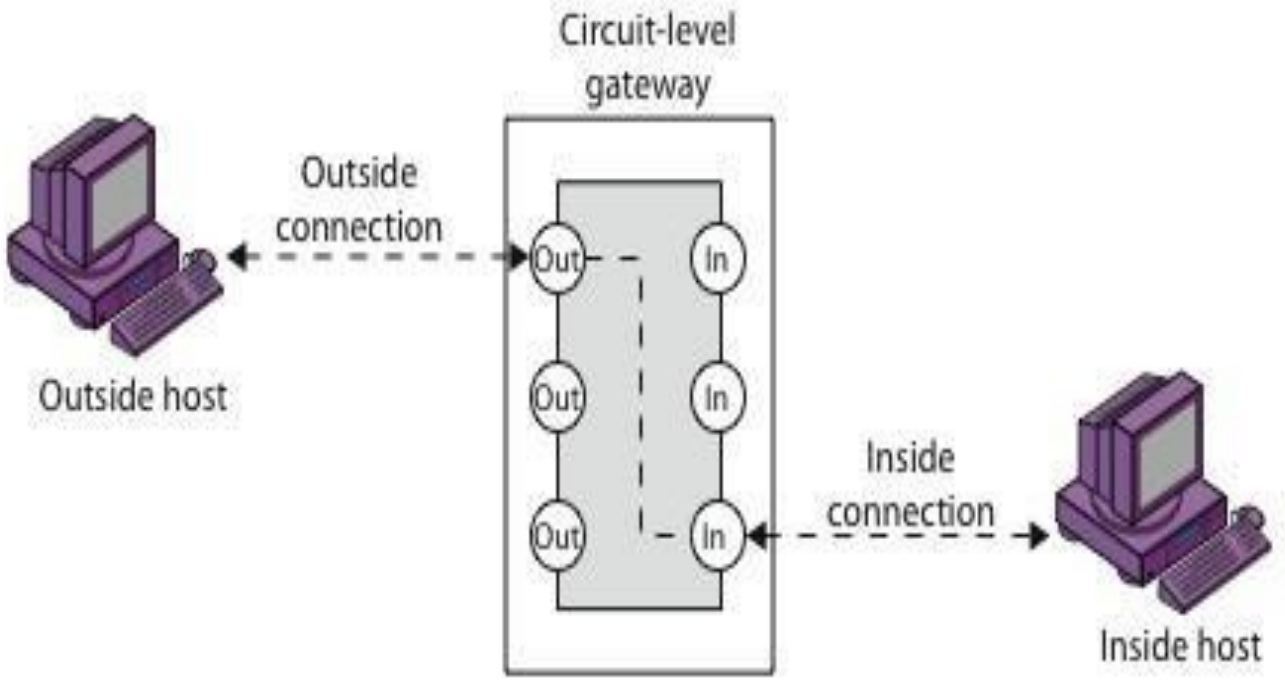



(b) Application-level gateway

# Firewalls - Circuit Level Gateway

- Relays two TCP connections
- Imposes security by limiting which such connections are allowed
- Once created usually relays traffic without examining contents
- Typically used when trust internal users by allowing general outbound connections
- SOCKS is commonly used

# Firewalls - Circuit Level Gateway

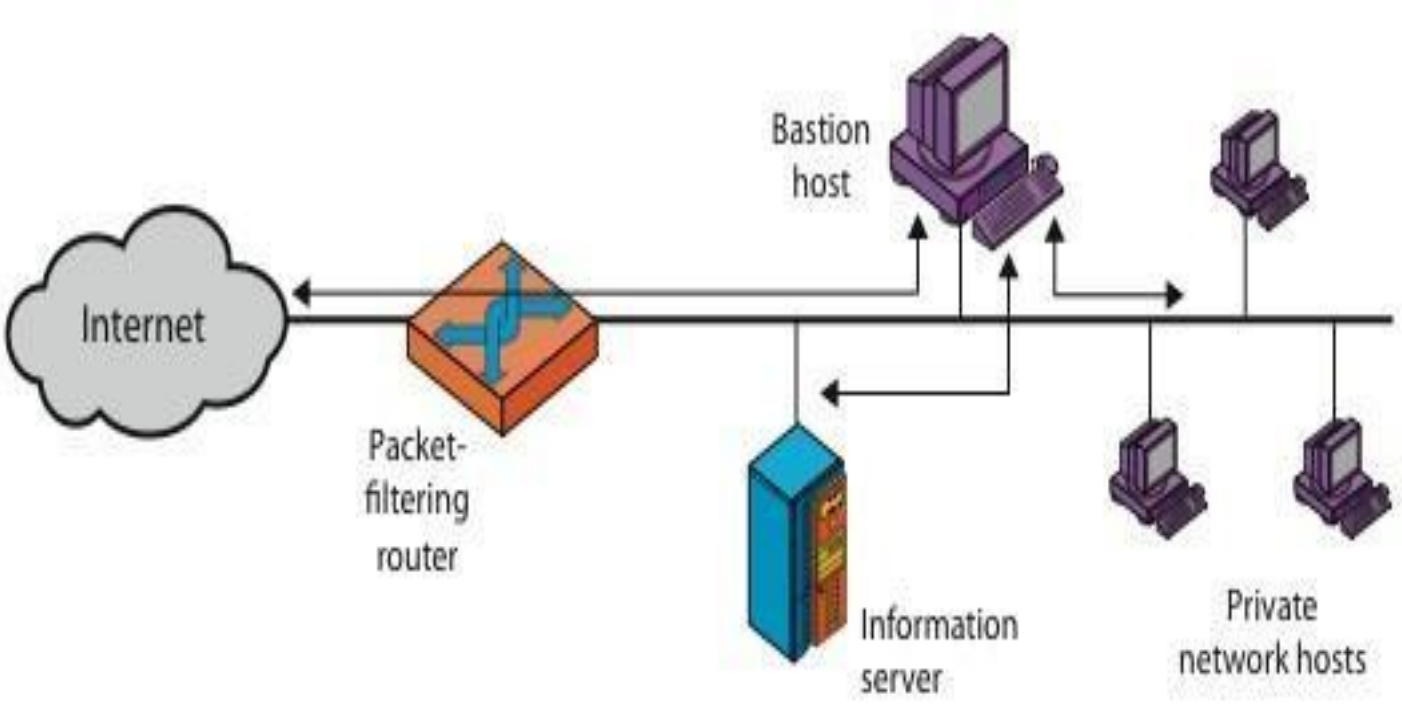



(c) Circuit-level gateway

# Bastion Host

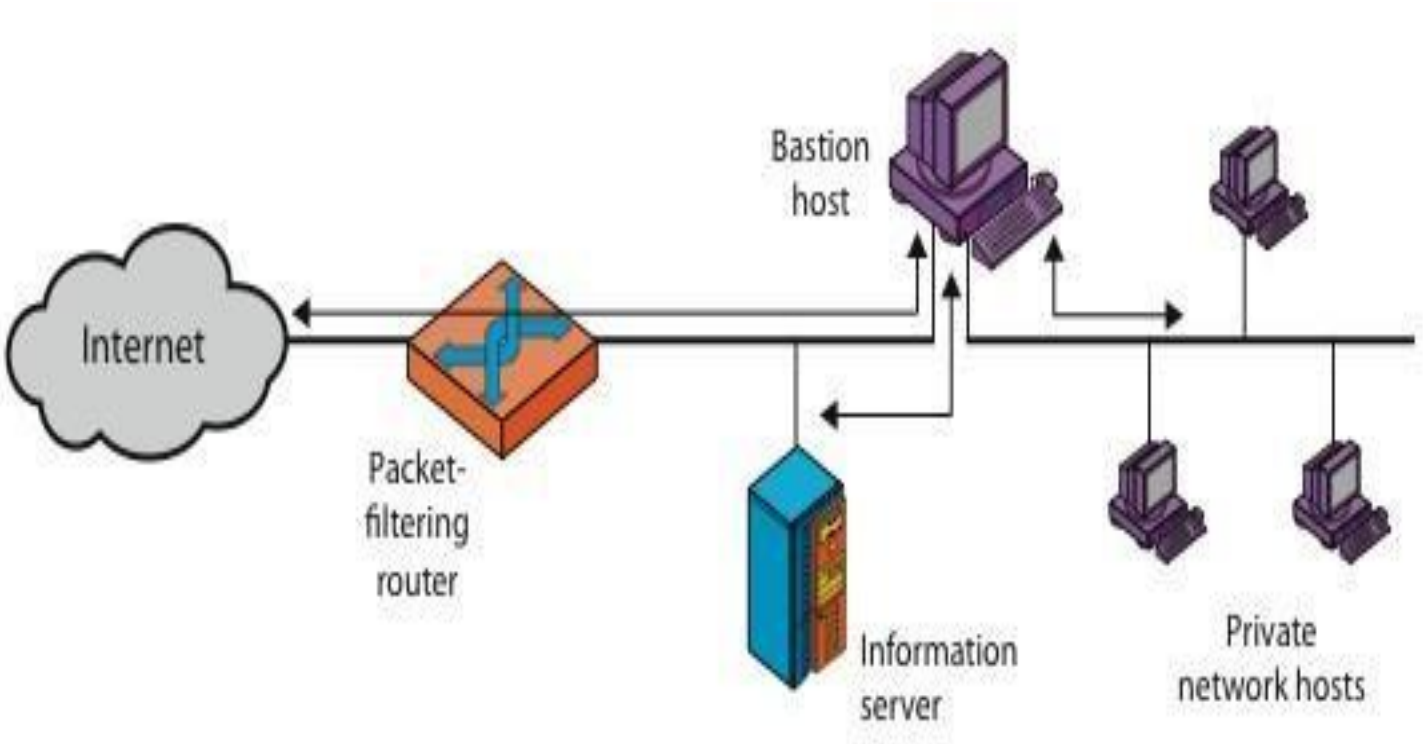
- Highly secure host system
- Runs circuit / application level gateways
- or provides externally accessible services
- Potentially exposed to "hostile" elements
- Hence is secured to withstand this
  - hardened O/S, essential services, extra auth
  - proxies small, secure, independent, non-privileged
- May support 2 or more net connections
- May be trusted to enforce policy of trusted separation between these net connections

# Firewall Configurations



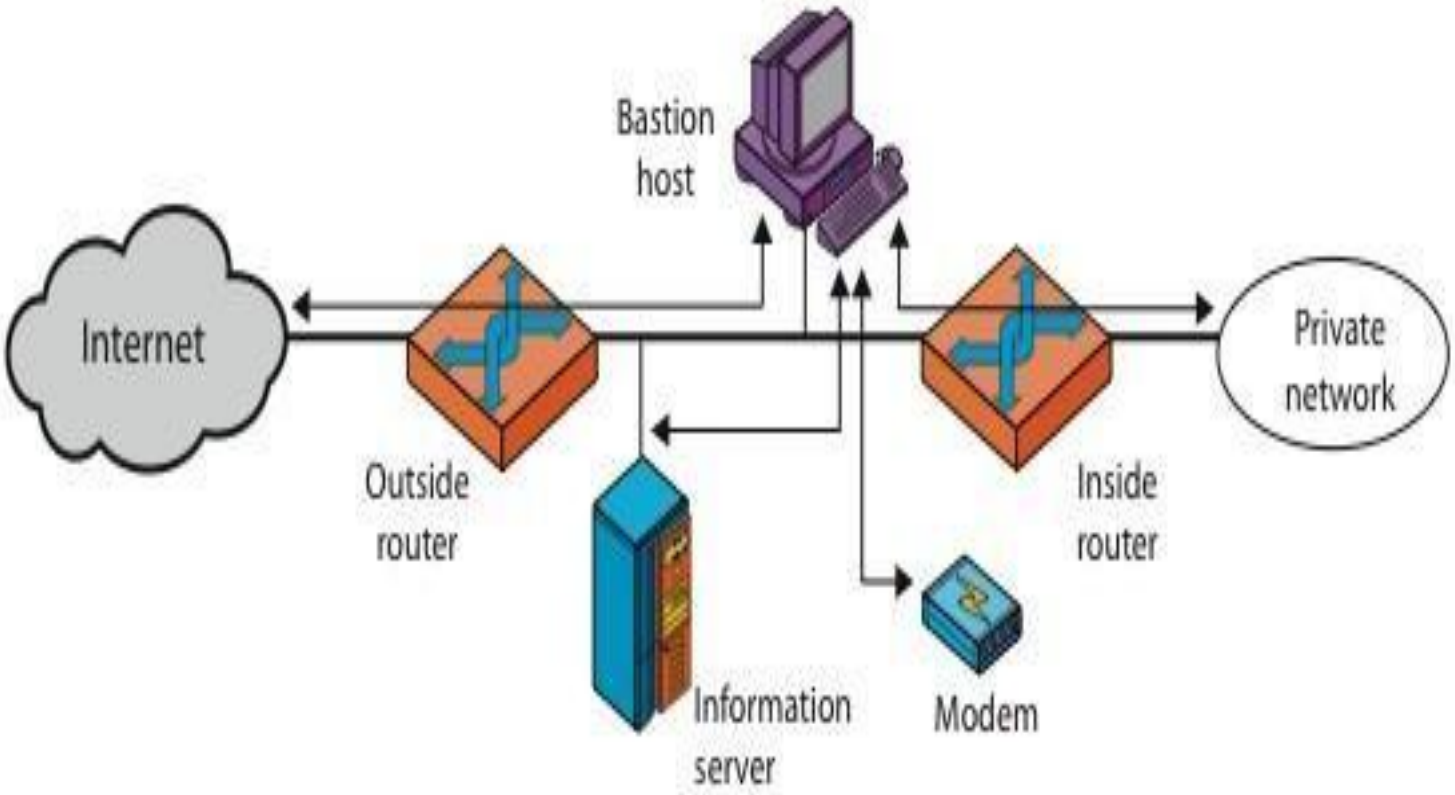

(a) Screened host firewall system (single-homed bastion host)

# Firewall Configurations




(b) Screened host firewall system (dual-homed bastion host)

# Firewall Configurations

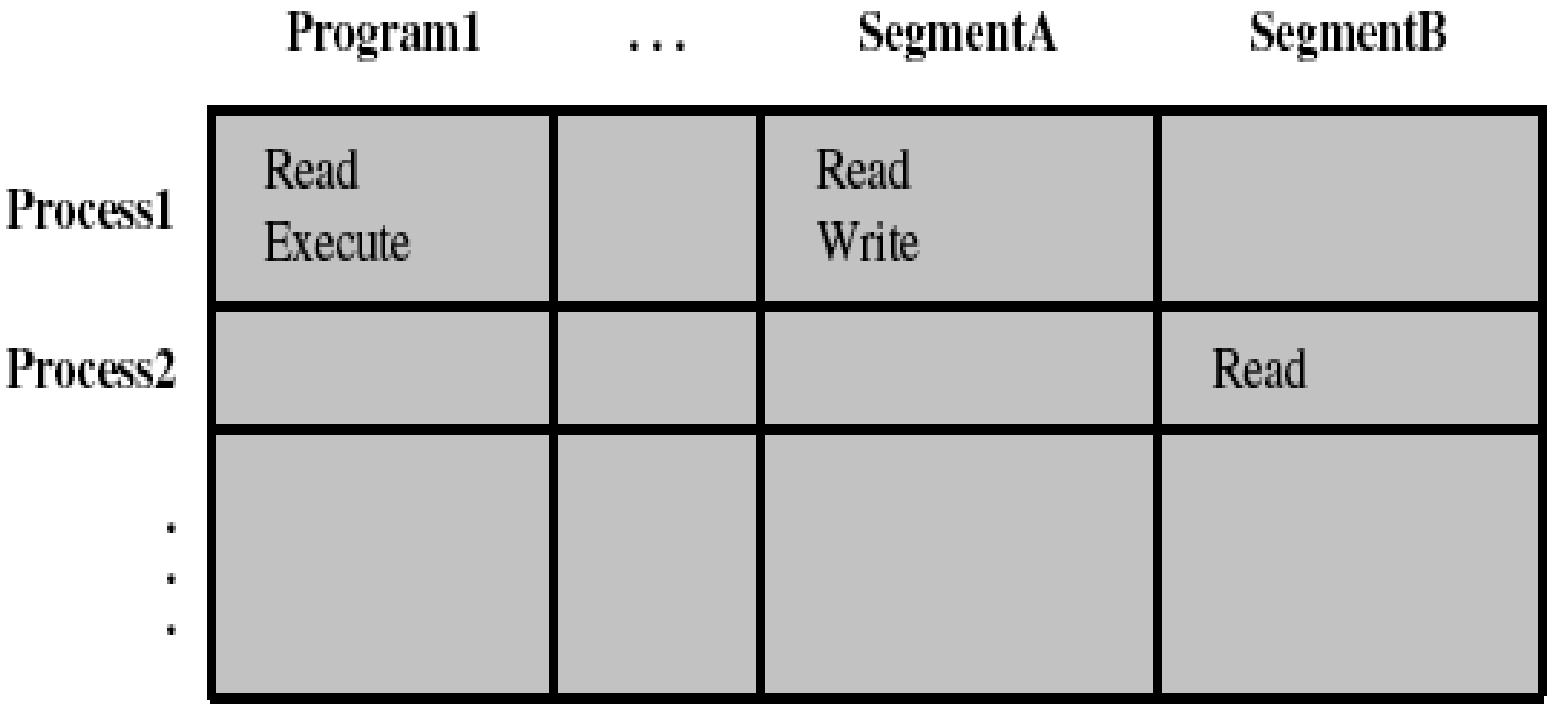



(c) Screened-subnet firewall system

# Access Control

- Given system has identified a user
- Determine what resources they can access
- General model is that of access matrix with
  - subject - active entity (user, process)
  - object - passive entity (file or resource)
  - access right – way object can be accessed
- Can decompose by
  - columns as access control lists
  - rows as capability tickets

# Access Control Matrix



| | Program1 | . . . | SegmentA | SegmentB |
|---|---|---|---|---|
| Process1 | Read<br>Execute | | Read<br>Write | |
| Process2 | | | | Read |
| . | | | | |
| . | | | | |
| . | | | | |

(a) Access matrix

# Trusted Computer Systems

- Information security is increasingly important
- Have varying degrees of sensitivity of information
  - cf military info classifications: confidential, secret etc
- Subjects (people or programs) have varying rights of access to objects (information)
- Known as multilevel security
  - subjects have maximum & current security level
  - objects have a fixed security level classification

# ThankYou