

Cryptography and Network Security

Third Edition
by William Stallings

Chapter 1 – Introduction

The art of war teaches us to rely not on the likelihood of the enemy's not coming, but on our own readiness to receive him; not on the chance of his not attacking, but rather on the fact that we have made our position unassailable.

—*The Art of War, Sun Tzu*

Background

- Information Security requirements have changed in recent times
- traditionally provided by physical and administrative mechanisms
- computer use requires automated tools to protect files and other stored information
- use of networks and communications links requires measures to protect data during transmission

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

Aim of Course

- our focus is on **Internet Security**
- consists of measures to deter, prevent, detect, and correct security violations that involve the transmission of information

Services, Mechanisms, Attacks

- need systematic way to define requirements
- consider three aspects of information security:
 - **security attack** (any action that compromises the security of info. Owned by org.)
 - **security mechanism** (is to detect, recover and prevent security attack)
 - **security service** (an enhancement to the security of data processing)

Security Service

- is something that enhances the security of the data processing systems and the information transfers of an organization
- intended to counter security attacks
- make use of one or more security mechanisms to provide the service
- replicate functions normally associated with physical documents
 - eg. have signatures, dates; need protection from disclosure, tampering, or destruction; be notarized or witnessed; be recorded or licensed

Security Mechanism

- a mechanism that is designed to detect, prevent, or recover from a security attack
- no single mechanism that will support all functions required
- however one particular element underlies many of the security mechanisms in use:
cryptographic techniques
- hence our focus on this area

Security Attack_(table 1.2,1.3)

- 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
- have a wide range of attacks
- can focus of generic types of attacks
- note: often *threat* & *attack* mean same

OSI Security Architecture

- ITU-T X.800 Security Architecture for OSI
- defines a systematic way of defining and providing security requirements
- for us it provides a useful, if abstract, overview of concepts we will study

Security Services

- X.800 defines it as: a service provided by a protocol layer of communicating open systems, which ensures adequate security of the systems or of data transfers
- RFC 2828 defines it as: a processing or communication service provided by a system to give a specific kind of protection to system resources
- X.800 defines it in 5 major categories

Security Services (X.800-table 1.4)

- **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

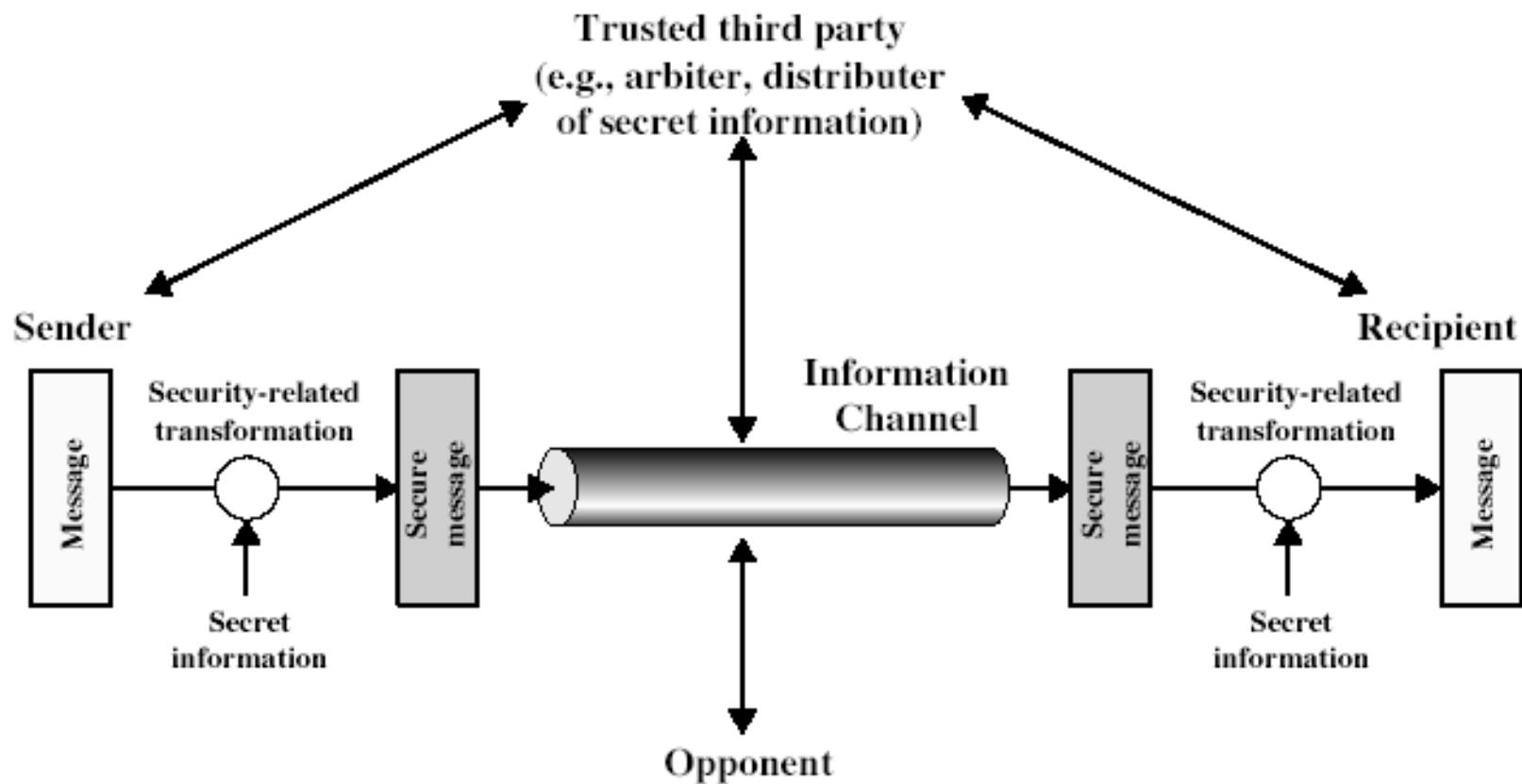
Security Mechanisms (X.800)

- 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

Classify Security Attacks as

- **passive attacks** - eavesdropping on, or monitoring of, transmissions to:
 - obtain message contents, or
 - monitor traffic flows
- **active attacks** – modification of data stream to:
 - masquerade of one entity as some other
 - replay previous messages
 - modify messages in transit
 - denial of service

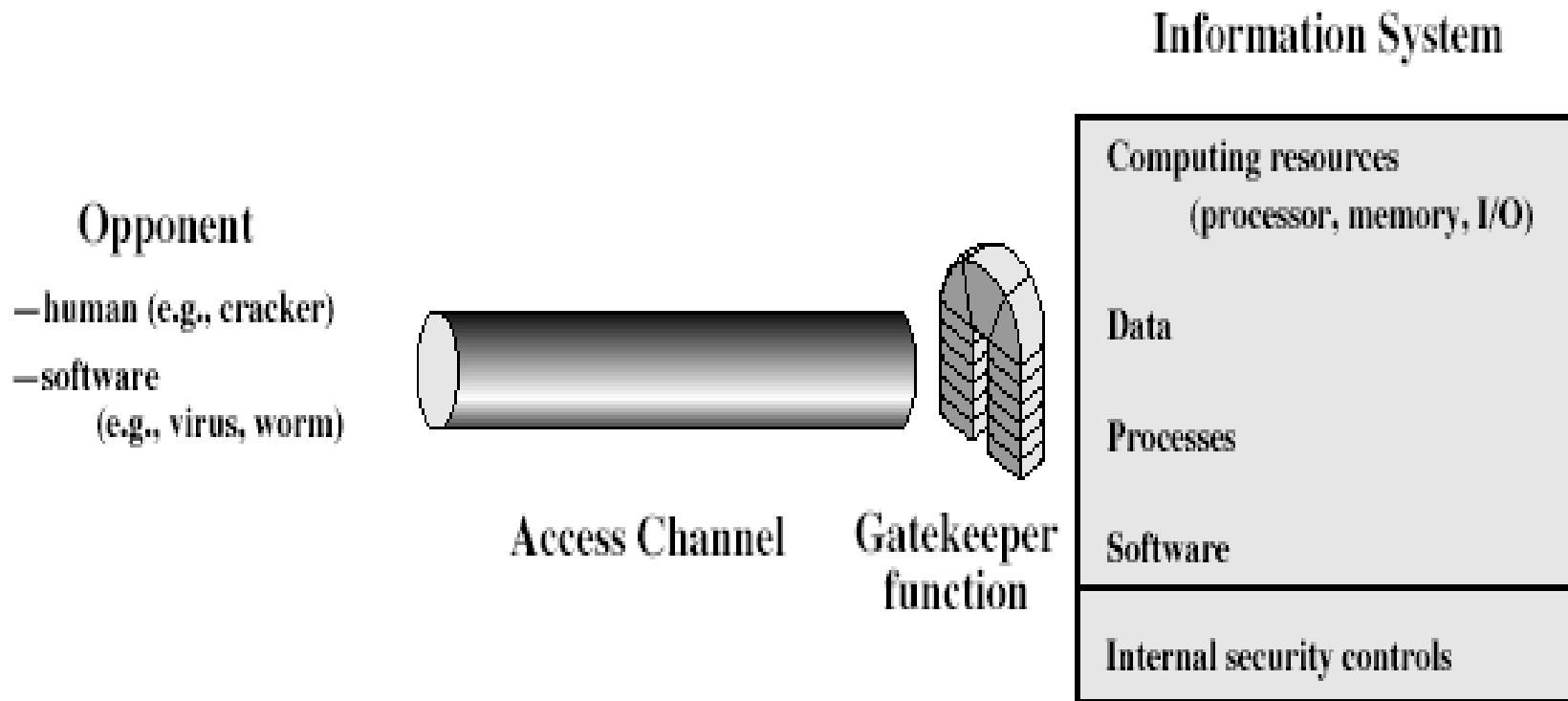
Model for Network Security



Model for Network Security

- using this model requires us to:
 - design a suitable algorithm for the security transformation
 - generate the secret information (keys) used by the algorithm
 - develop methods to distribute and share the secret information
 - specify a protocol enabling the principals to use the transformation and secret information for a security service

Model for Network Access Security



Model for Network Access Security

- using this model requires us to:
 - select appropriate gatekeeper functions to identify users
 - implement security controls to ensure only authorised users access designated information or resources
- trusted computer systems can be used to implement this model

Summary

- have considered:
 - computer, network, internet security def's
 - security services, mechanisms, attacks
 - X.800 standard
 - models for network (access) security

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 2 – Classical Encryption Techniques

*Many savages at the present day regard their names as vital parts of themselves, and therefore take great pains to conceal their real names, lest these should give to evil-disposed persons a handle by which to injure their owners. —**The Golden Bough, Sir James George Frazer***

Symmetric Encryption

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

Basic Terminology

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

Requirements

- two requirements for secure use of symmetric encryption:
 - a strong encryption algorithm
 - a secret key known only to sender / receiver
$$Y = E_K(X)$$
$$X = D_K(Y)$$
- assume encryption algorithm is known
- implies a secure channel to distribute key

Cryptography

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

Types of Cryptanalytic Attacks

- **ciphertext only**
 - only know algorithm / ciphertext, statistical, can identify plaintext
- **known plaintext**
 - know/suspect plaintext & ciphertext to attack cipher
- **chosen plaintext**
 - select plaintext and obtain ciphertext to attack cipher
- **chosen ciphertext**
 - select ciphertext and obtain plaintext to attack cipher
- **chosen text**
 - select either plaintext or ciphertext to en/decrypt to attack cipher

Brute Force Search

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

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

More Definitions

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

Classical Substitution Ciphers

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

Caesar Cipher

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

meet me after the toga party

PHHW PH DIWHU WKH WRJD SDUWB

Caesar Cipher

- can define transformation as:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

- mathematically give each letter a number

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12
n	o	p	q	r	s	t	u	v	w	x	y	z
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: if we wish to replace letters

Ciphertext: WIRFRWAJUHYFTSDVFSUUUFYA

Monoalphabetic Cipher Security

- now have a total of $26! = 4 \times 10^{26}$ keys
- with so many keys, might think is secure
- but would be **!!!WRONG!!!**
- problem is language characteristics

Language Redundancy and Cryptanalysis

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

Use in Cryptanalysis

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

Example Cryptanalysis

- given ciphertext:

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUBMETSXAIZ
VUEPHZHMDZSHZOWSFAPPDTSPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

- count relative letter frequencies (see text)
- guess P & Z are e and t
- guess ZW is th and hence ZWP is the
- proceeding with trial and error finally get:

it was disclosed yesterday that several informal but direct contacts have been made with political representatives of the viet cong in moscow

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

Playfair 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

MONAR

CHYBD

EFGIK

LPQST

UVWXZ

Encrypting and Decrypting

- plaintext encrypted two letters at a time:
 1. if a pair is a repeated letter, insert a filler like 'X', eg. "balloon" encrypts as "ba lx lo on"
 2. if both letters fall in the same row, replace each with letter to right (wrapping back to start from end), eg. "ar" encrypts as "RM"
 3. if both letters fall in the same column, replace each with the letter below it (again wrapping to top from bottom), eg. "mu" encrypts to "CM"
 4. otherwise each letter is replaced by the one in its row in the column of the other letter of the pair, eg. "hs" encrypts to "BP", and "ea" to "IM" or "JM" (as desired)

Security of the Playfair Cipher

- security much improved over monoalphabetic
- since have $26 \times 26 = 676$ digrams
- 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. US & British military in WW1)
- it **can** be broken, given a few hundred letters
- since still has much of plaintext structure

Polyalphabetic Ciphers

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

Vigenère Cipher

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

Example

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

key: deceptive deceptive deceptive

plaintext: wearediscoveredsaveyourself

ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMJ

Aids

- simple aids can assist with en/decryption
- a **Saint-Cyr Slide** is a simple manual aid
 - a slide with repeated alphabet
 - line up plaintext 'A' with key letter, eg 'C'
 - then read off any mapping for key letter
- can bend round into a **cipher disk**
- or expand into a **Vigenère Tableau** (see text Table 2.3)

Security of Vigenère Ciphers

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

Autokey Cipher

- ideally want a key as long as the message
- Vigenère proposed the **autokey** cipher
- with keyword is prefixed to message as key
- knowing keyword can recover the first few letters
- use these in turn on the rest of the message
- but still have frequency characteristics to attack
- eg. given key *deceptive*

key: deceptivewearediscoveredsav

plaintext: wearediscoveredsaveyourself

ciphertext: ZICVTWQNGKZEIIGASXSTSLVVWLA

One-Time Pad

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

H	E	L	L	O	message
7 (H)	4 (E)	11 (L)	11 (L)	14 (O)	message
+ 23 (X)	12 (M)	2 (C)	10 (K)	11 (L)	key
= 30	16	13	21	25	message + key
= 4 (E)	16 (Q)	13 (N)	21 (V)	25 (Z)	message + key (mod 26)
	E	Q	N	V	Z → ciphertext

E	Q	N	V	Z	ciphertext
4 (E)	16 (Q)	13 (N)	21 (V)	25 (Z)	ciphertext
- 23 (X)	12 (M)	2 (C)	10 (K)	11 (L)	key
= -19	4	11	11	14	ciphertext - key
= 7 (H)	4 (E)	11 (L)	11 (L)	14 (O)	ciphertext - key (mod 26)
	H	E	L	L	O → message

Transposition Ciphers

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

Rail Fence cipher

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

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

- giving ciphertext

MEMATRHTGPRYETEFETEOAAT

Row Transposition Ciphers

- a more complex scheme
- 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: TTNAAPMTSUOAODWCOIXKNLYPETZ

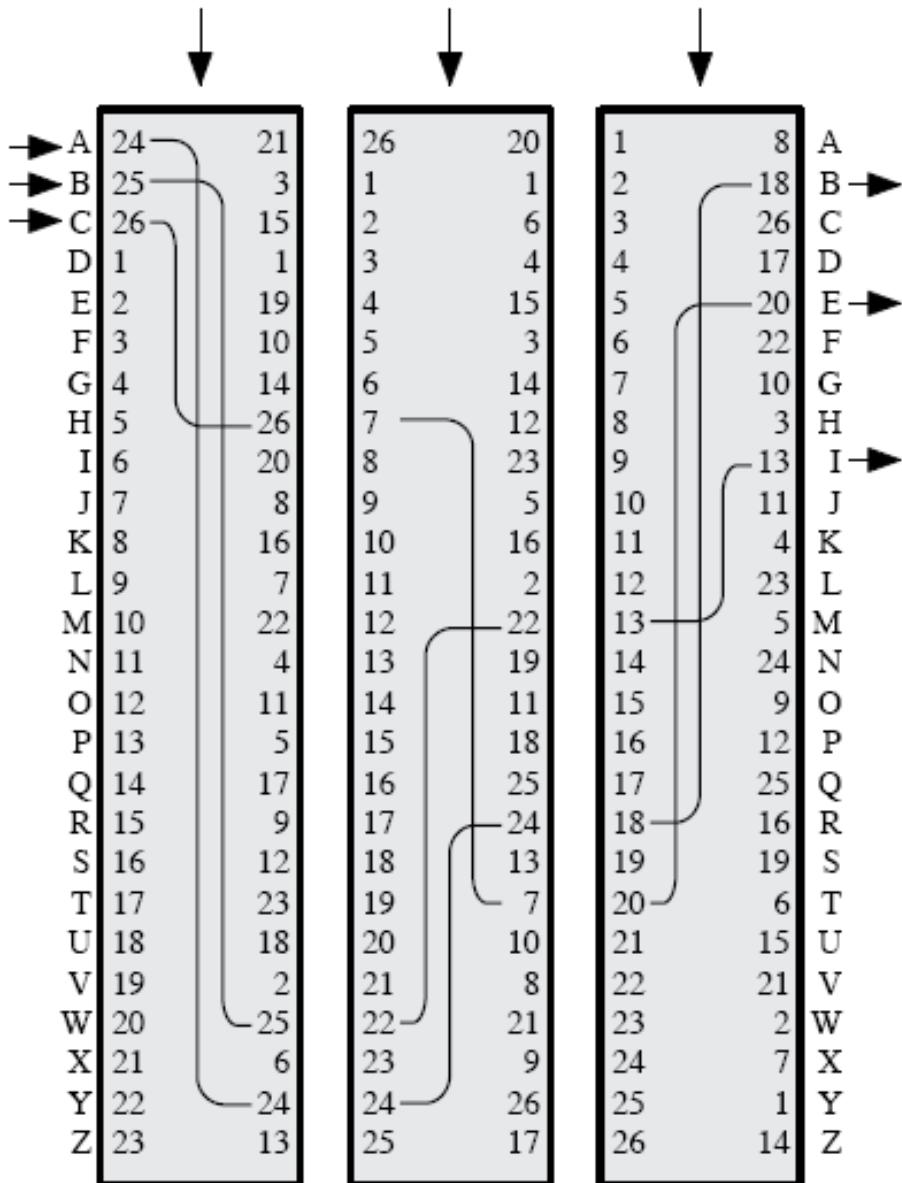
Product Ciphers

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

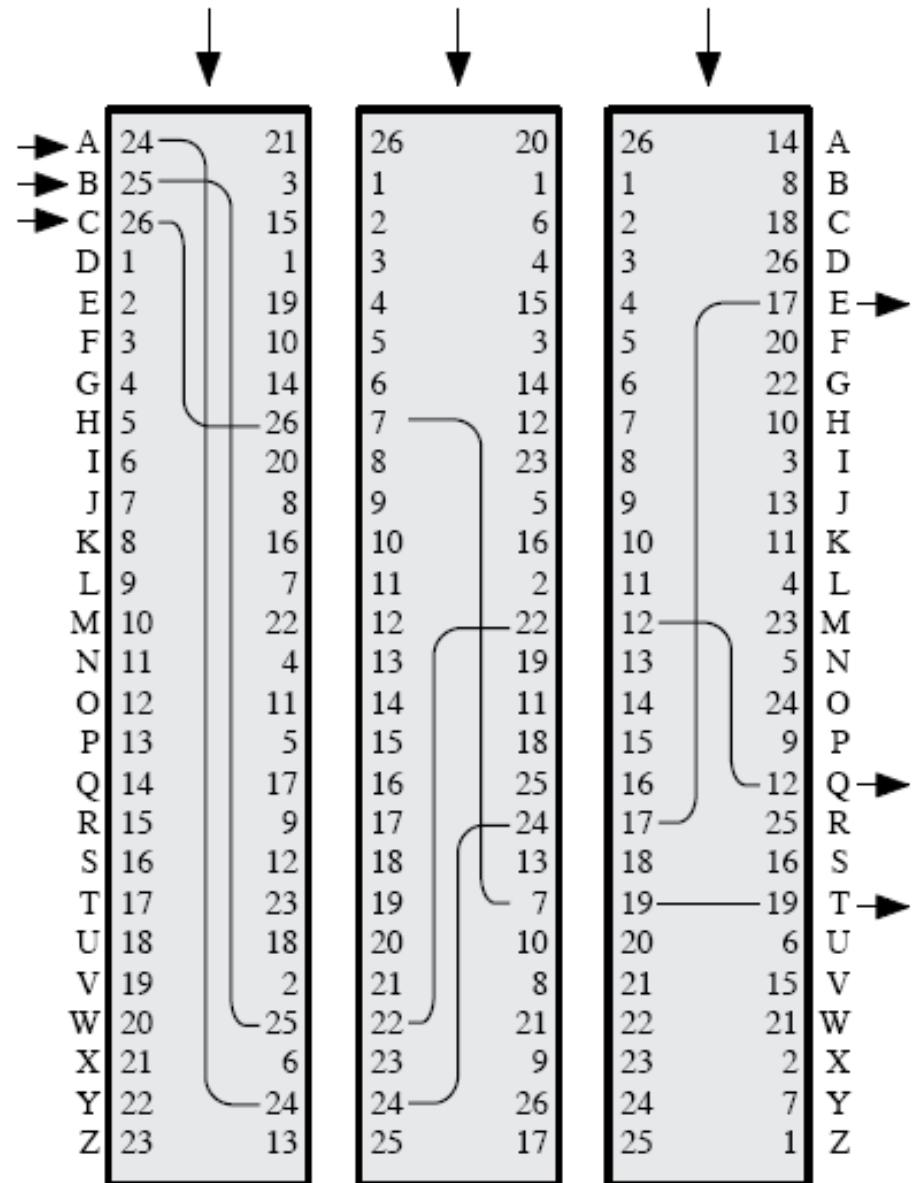
Rotor Machines

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

direction of motion



direction of motion



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

Summary

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

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 3 – Block Ciphers and the Data Encryption Standard

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

—***Talking to Strange Men, Ruth Rendell***

Modern Block Ciphers

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

Block vs Stream Ciphers

- block ciphers process messages in blocks, each of which is then en/decrypted
- like a substitution on very big characters
 - 64-bits or more
- stream ciphers process messages a bit or byte at a time when en/decrypting
- many current ciphers are block ciphers
- hence are focus of course

Block Cipher Principles

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

Claude Shannon and Substitution-Permutation Ciphers

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

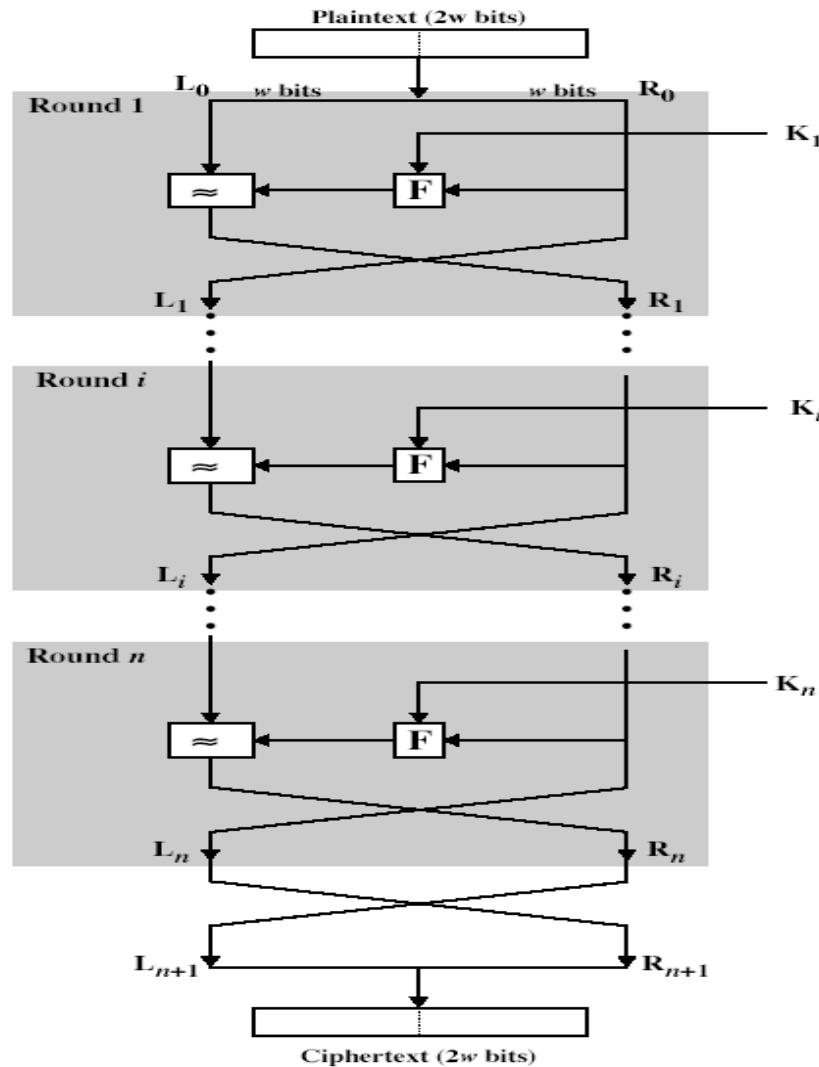
Confusion and Diffusion

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

Feistel Cipher Structure

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

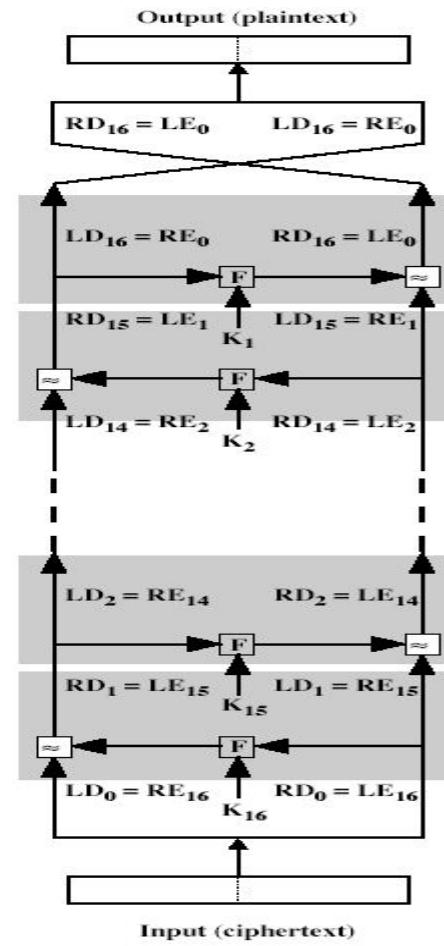
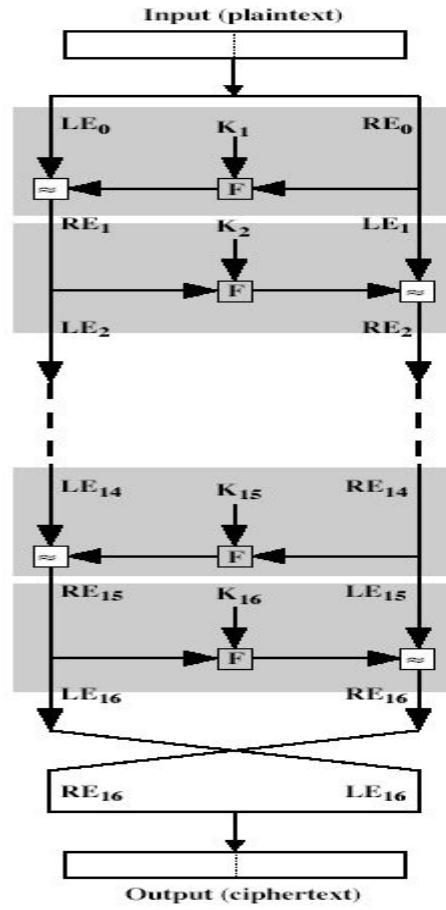
Feistel Cipher Structure



Feistel Cipher Design Principles

- **block size**
 - increasing size improves security, but slows cipher
- **key size**
 - increasing size improves security, makes exhaustive key searching harder, but may slow cipher
- **number of rounds**
 - increasing number improves security, but slows cipher
- **subkey generation**
 - greater complexity can make analysis harder, but slows cipher
- **round function**
 - greater complexity can make analysis harder, but slows cipher
- **fast software en/decryption & ease of analysis**
 - are more recent concerns for practical use and testing

Feistel Cipher Decryption



Data Encryption Standard (DES)

- most widely used block cipher in world
- adopted in 1977 by NBS (now NIST)
 - as FIPS PUB 46
- encrypts 64-bit data using 56-bit key
- has widespread use
- has been considerable controversy over its security

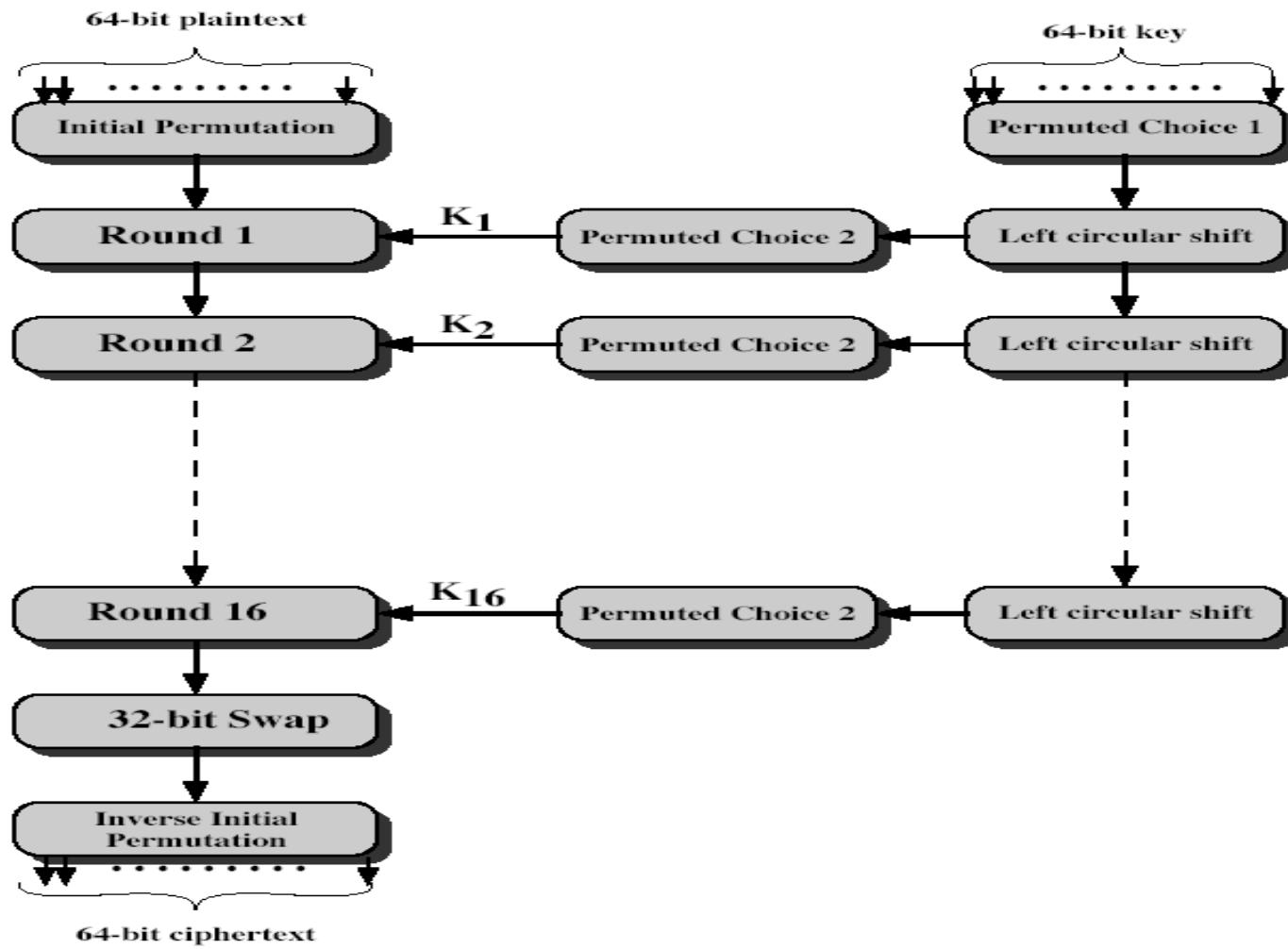
DES History

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

DES Design Controversy

- although DES standard is public
- was considerable controversy over design
 - in choice of 56-bit key (vs Lucifer 128-bit)
 - and because design criteria were classified
- subsequent events and public analysis show in fact design was appropriate
- DES has become widely used, esp in financial applications

DES Encryption



Initial Permutation IP

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

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

DES Round Structure

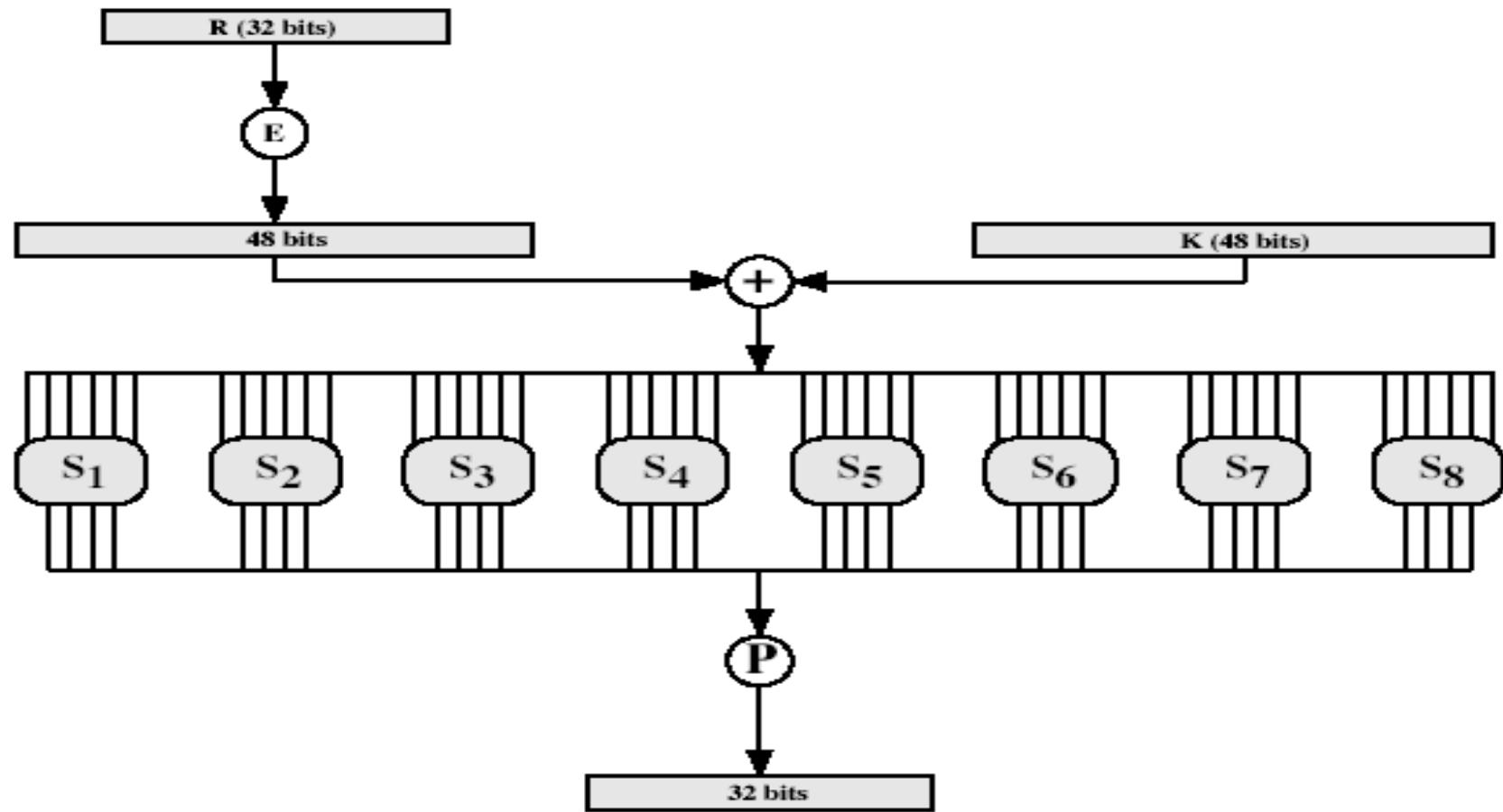
- uses two 32-bit L & R halves
- as for any Feistel cipher can describe as:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \text{ xor } F(R_{i-1}, K_i)$$

- takes 32-bit R half and 48-bit subkey and:
 - expands R to 48-bits using perm E
 - adds to subkey
 - passes through 8 S-boxes to get 32-bit result
 - finally permutes this using 32-bit perm P

DES Round Structure



Substitution Boxes S

- have eight S-boxes which map 6 to 4 bits
- each S-box is actually 4 little 4 bit boxes
 - outer bits 1 & 6 (**row** bits) select one rows
 - inner bits 2-5 (**col** bits) are substituted
 - result is 8 lots of 4 bits, or 32 bits
- row selection depends on both data & key
 - feature known as autoclaving (autokeying)
- example:

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

DES Key Schedule

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

DES Decryption

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

Avalanche Effect

- key desirable property of encryption alg
- where a change of **one** input or key bit results in changing approx **half** output bits
- making attempts to “home-in” by guessing keys impossible
- DES exhibits strong avalanche

Strength of DES – Key Size

- 56-bit keys have $2^{56} = 7.2 \times 10^{16}$ values
- brute force search looks hard
- recent advances have shown is possible
 - in 1997 on Internet in a few months
 - in 1998 on dedicated h/w (EFF) in a few days
 - in 1999 above combined in 22hrs!
- still must be able to recognize plaintext
- now considering alternatives to DES

Strength of DES – Timing Attacks

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

Strength of DES – Analytic Attacks

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

Differential Cryptanalysis

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

Differential Cryptanalysis

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

Differential Cryptanalysis

Compares Pairs of Encryptions

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

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

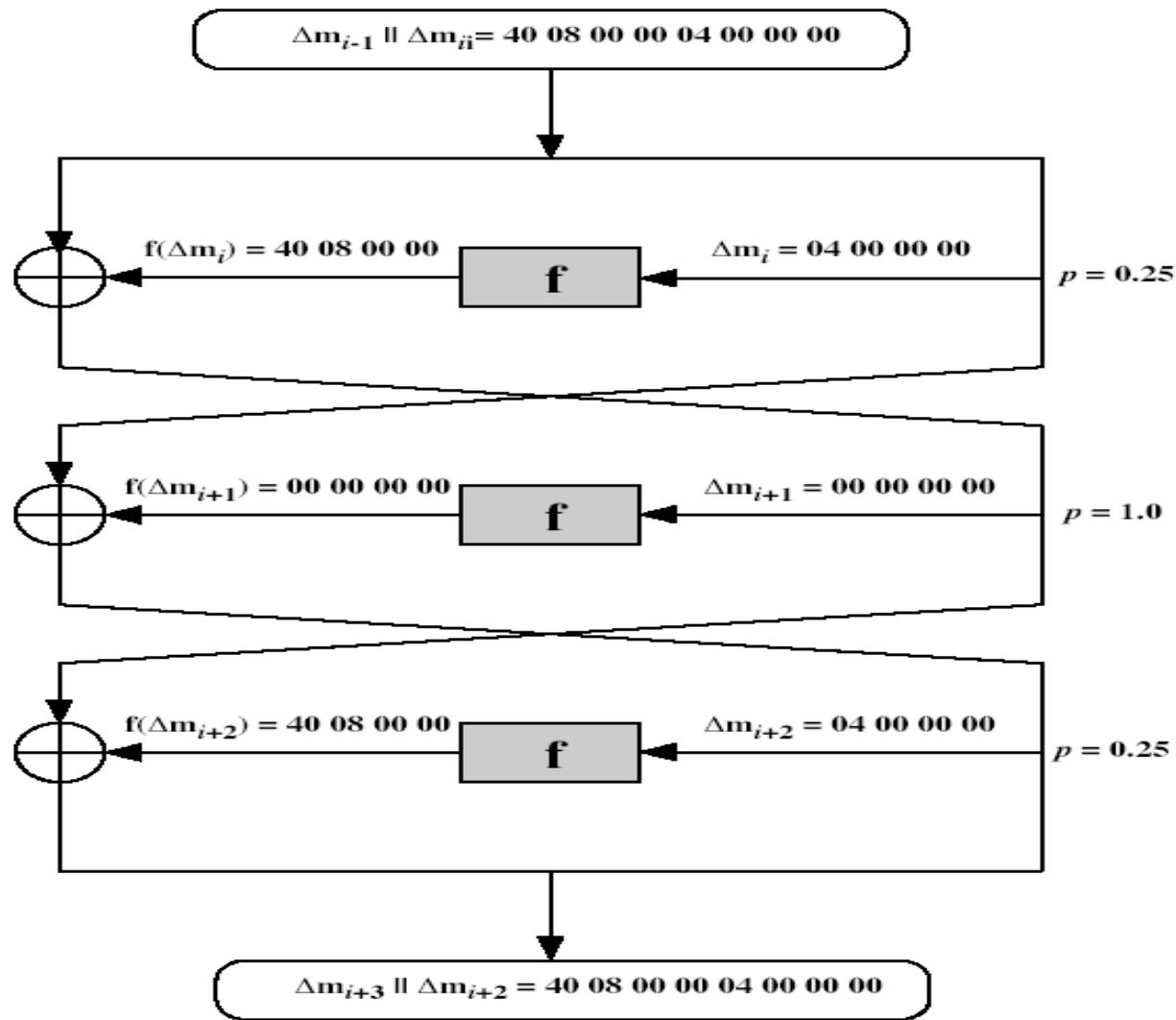
$$= [m_{i-1} \oplus f(m_i, K_i)] \oplus [m'_{i-1} \oplus f(m'_i, K_i)]$$

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

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^{47} known plaintexts, still in practise infeasible

Linear Cryptanalysis

- find linear approximations with prob $p \neq \frac{1}{2}$
$$P[i_1, i_2, \dots, i_a] (+) C[j_1, j_2, \dots, j_b] = K[k_1, k_2, \dots, k_c]$$
where i_a, j_b, k_c 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: $|p - \frac{1}{2}|$

Block Cipher Design Principles

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

Modes of Operation

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

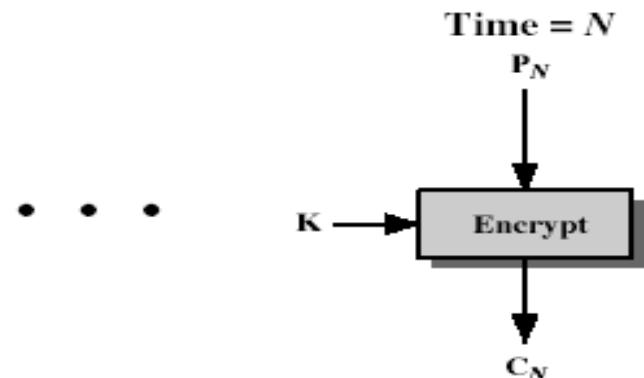
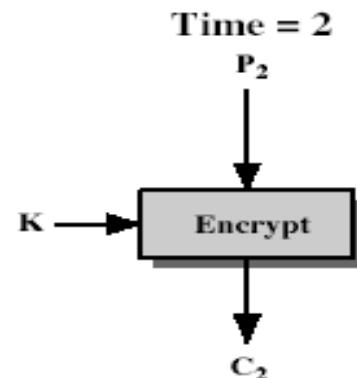
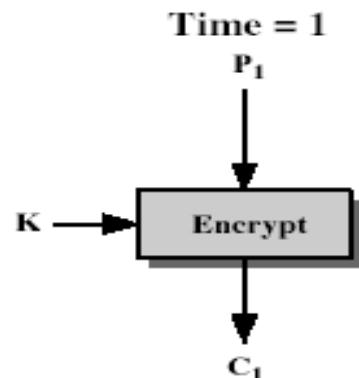
Electronic Codebook Book (ECB)

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

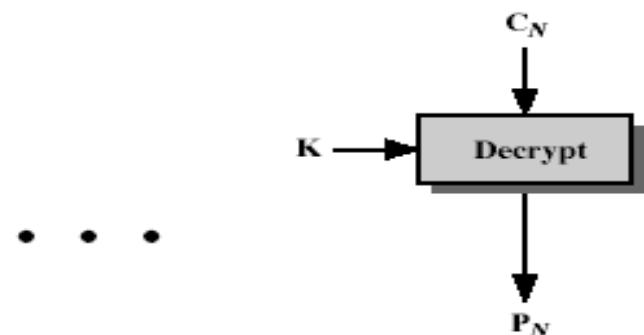
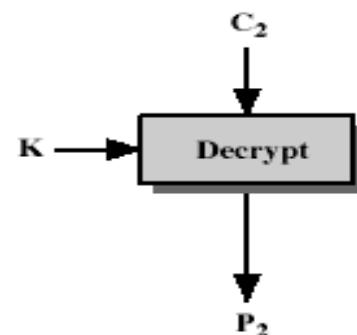
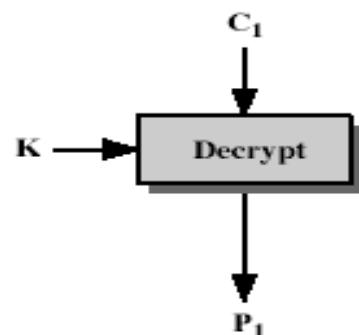
$$C_i = \text{DES}_{K1}(P_i)$$

- uses: secure transmission of single values

Electronic Codebook Book (ECB)



(a) Encryption



(b) Decryption

Advantages and Limitations of ECB

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

Cipher Block Chaining (CBC)

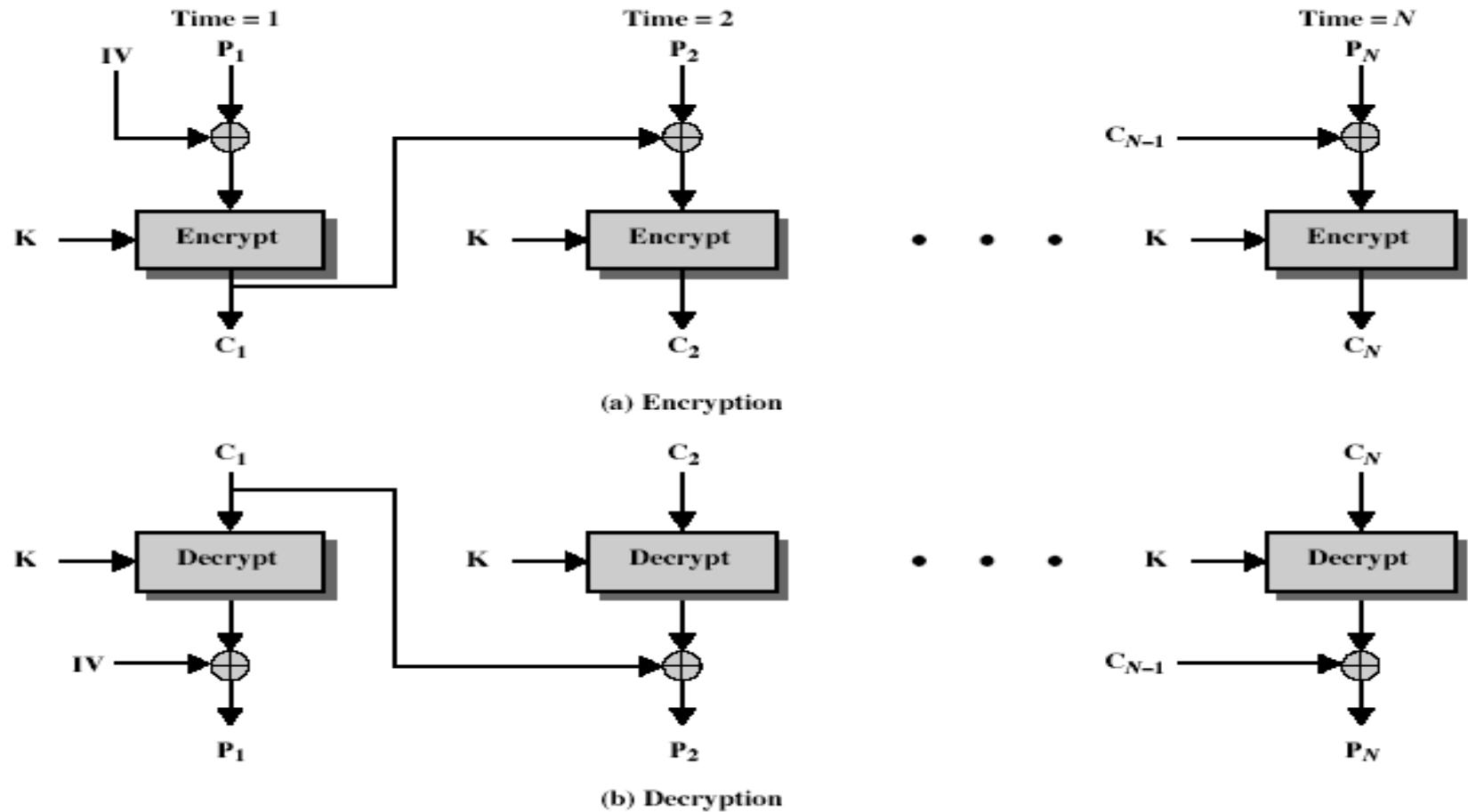
- message is broken into blocks
- but these are linked together in the encryption operation
- each previous cipher blocks is chained with current plaintext block, hence name
- use Initial Vector (IV) to start process

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

$$C_{-1} = \text{IV}$$

- uses: bulk data encryption, authentication

Cipher Block Chaining (CBC)



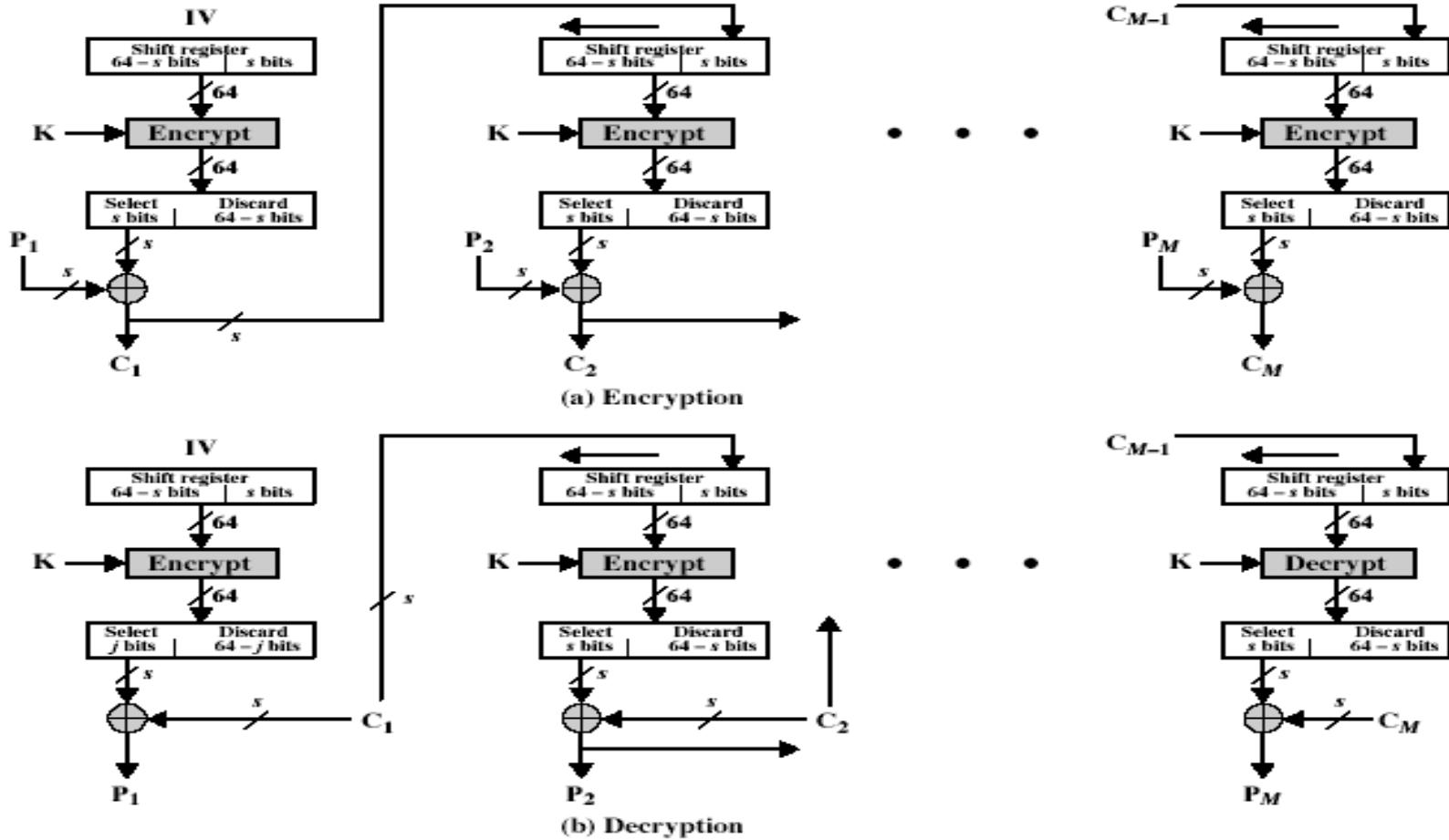
Advantages and Limitations of CBC

- each ciphertext block depends on **all** message blocks
- thus a change in the message affects all ciphertext blocks after the change as well as the original block
- need **Initial Value (IV)** known to sender & receiver
 - however if IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate
 - hence either IV must be a fixed value (as in EFTPOS) or it must be sent encrypted in ECB mode before rest of message
- at end of message, handle possible last short block
 - by padding either with known non-data value (eg nulls)
 - or pad last block with count of pad size
 - eg. [b1 b2 b3 0 0 0 0 5] <- 3 data bytes, then 5 bytes pad+count

Cipher FeedBack (CFB)

- message is treated as a stream of bits
- added to the output of the block cipher
- result is feed back for next stage (hence name)
- standard allows any number of bit (1,8 or 64 or whatever) to be feed back
 - denoted CFB-1, CFB-8, CFB-64 etc
- is most efficient to use all 64 bits (CFB-64)
$$C_i = P_i \text{ XOR } DES_{K1}(C_{i-1})$$
$$C_{-1} = IV$$
- uses: stream data encryption, authentication

Cipher FeedBack (CFB)



Advantages and Limitations of CFB

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

Output FeedBack (OFB)

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

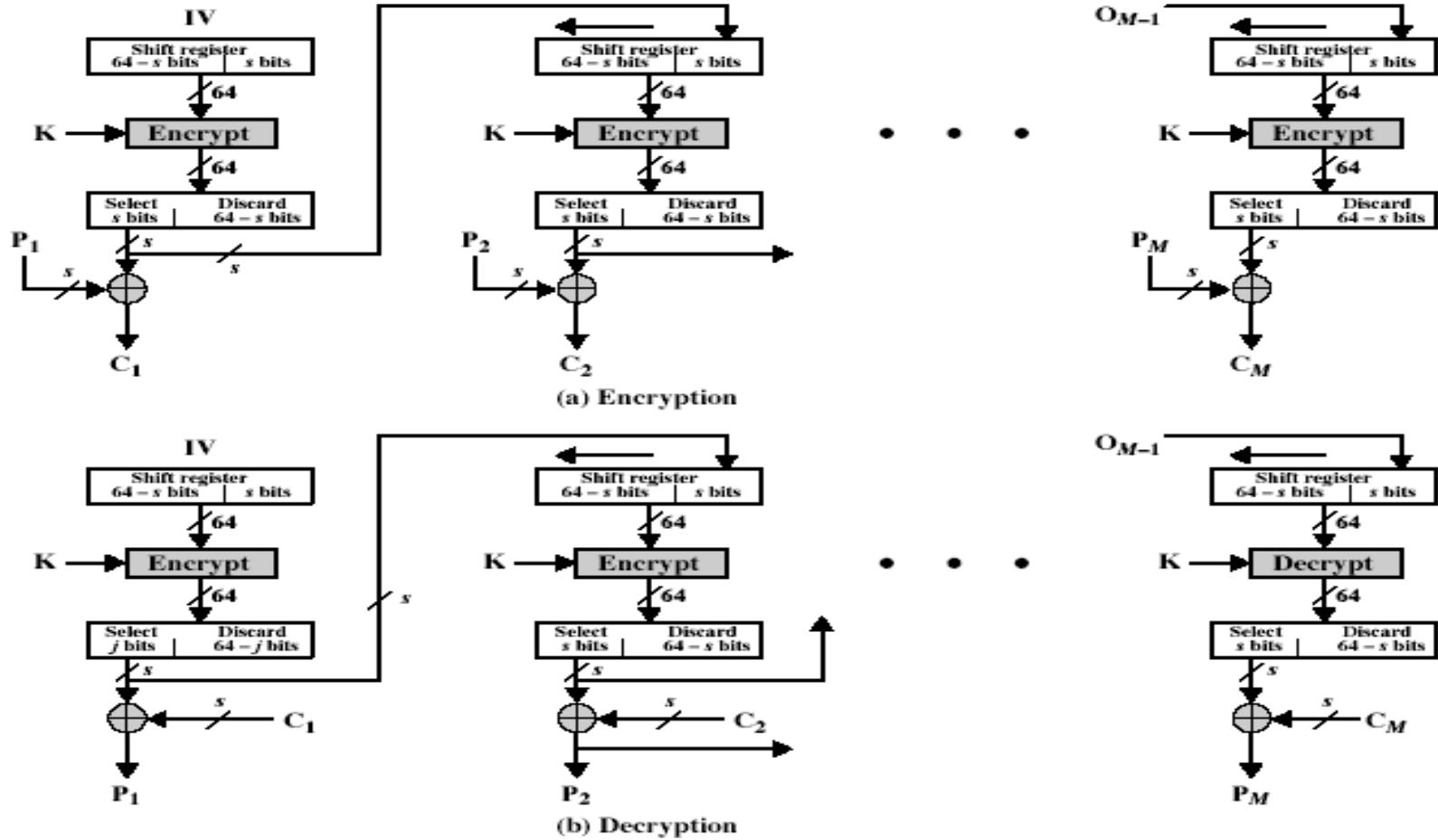
$$C_i = P_i \text{ XOR } O_i$$

$$O_i = \text{DES}_{K1}(O_{i-1})$$

$$O_{-1} = \text{IV}$$

- uses: stream encryption over noisy channels

Output FeedBack (OFB)



Advantages and Limitations of OFB

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

Counter (CTR)

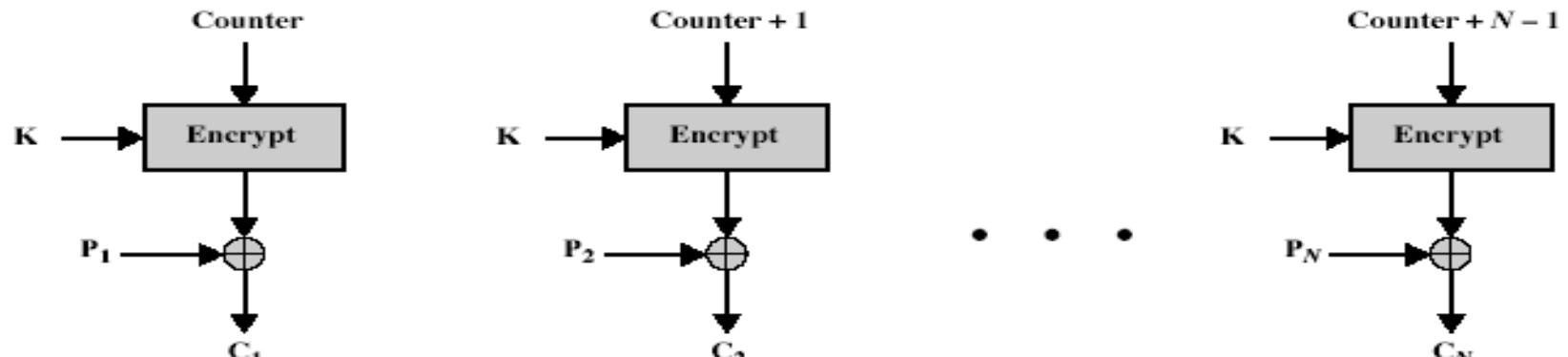
- a “new” mode, though proposed early on
- similar to OFB but encrypts counter value rather than any feedback value
- must have a different key & counter value for every plaintext block (never reused)

$$C_i = P_i \text{ XOR } O_i$$

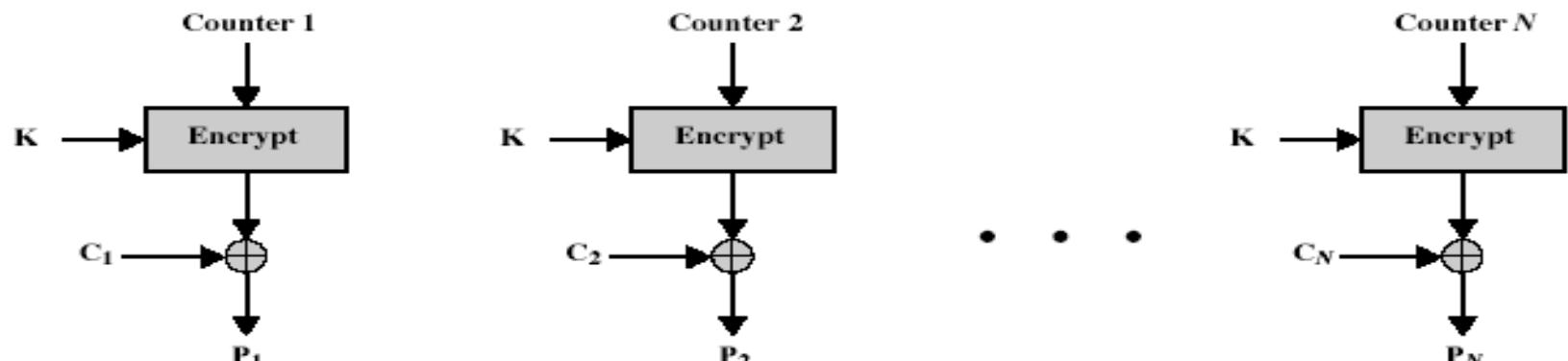
$$O_i = \text{DES}_{K1}(i)$$

- uses: high-speed network encryptions

Counter (CTR)



(a) Encryption



(b) Decryption

Advantages and Limitations of CTR

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

Summary

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

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 4 – Finite Fields

The next morning at daybreak, Star flew indoors, seemingly keen for a lesson. I said, "Tap eight." She did a brilliant exhibition, first tapping it in 4, 4, then giving me a hasty glance and doing it in 2, 2, 2, 2, before coming for her nut. It is astonishing that Star learned to count up to 8 with no difficulty, and of her own accord discovered that each number could be given with various different divisions, this leaving no doubt that she was consciously thinking each number. In fact, she did mental arithmetic, although unable, like humans, to name the numbers. But she learned to recognize their spoken names almost immediately and was able to remember the sounds of the names. Star is unique as a wild bird, who of her own free will pursued the science of numbers with keen interest and astonishing intelligence.

— ***Living with Birds, Len Howard***

Introduction

- will now introduce finite fields
- of increasing importance in cryptography
 - AES, Elliptic Curve, IDEA, Public Key
- concern operations on “numbers”
 - where what constitutes a “number” and the type of operations varies considerably
- start with concepts of groups, rings, fields from abstract algebra

Group

- a set of elements or “numbers”
- with some operation whose result is also in the set (closure)
- obeys:
 - associative law: $(a.b).c = a.(b.c)$
 - has identity e : $e.a = a.e = a$
 - has inverses a^{-1} : $a.a^{-1} = e$
- if commutative $a.b = b.a$
 - then forms an **abelian group**

Cyclic Group

- define **exponentiation** as repeated application of operator
 - example: $a^{-3} = a \cdot a \cdot a$
- and let identity be: $e=a^0$
- a group is cyclic if every element is a power of some fixed element
 - ie $b = a^k$ for some a and every b in group
- a is said to be a generator of the group

Ring

- a set of “numbers” with two operations (addition and multiplication) which are:
- an abelian group with addition operation
- multiplication:
 - has closure
 - is associative
 - distributive over addition: $a(b+c) = ab + ac$
- if multiplication operation is commutative, it forms a **commutative ring**
- if multiplication operation has inverses and no zero divisors, it forms an **integral domain**

Field

- a set of numbers with two operations:
 - abelian group for addition
 - abelian group for multiplication (ignoring 0)
 - ring

Modular Arithmetic

- define **modulo operator** $a \bmod n$ to be remainder when a is divided by n
- use the term **congruence** for: $a \equiv b \pmod{n}$
 - when divided by n , a & b have same remainder
 - eg. $100 = 34 \pmod{11}$
- b is called the **residue** of $a \bmod n$
 - since with integers can always write: $a = qn + b$
- usually have $0 \leq b \leq n-1$
 - $-12 \bmod 7 \equiv -5 \bmod 7 \equiv 2 \bmod 7 \equiv 9 \bmod 7$

Modulo 7 Example

...

-21 -20 -19 -18 -17 -16 -15

-14 -13 -12 -11 -10 -9 -8

-7 -6 -5 -4 -3 -2 -1

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 26 27

28 29 30 31 32 33 34

...

Divisors

- say a non-zero number b **divides** a if for some m have $a=mb$ (a, b, m all integers)
- that is b divides into a with no remainder
- denote this $b \mid a$
- and say that b is a **divisor** of a
- eg. all of 1,2,3,4,6,8,12,24 divide 24

Modular Arithmetic Operations

- is 'clock arithmetic'
- uses a finite number of values, and loops back from either end
- modular arithmetic is when do addition & multiplication and modulo reduce answer
- can do reduction at any point, ie
 - $a+b \bmod n = [a \bmod n + b \bmod n] \bmod n$

Modular Arithmetic

- can do modular arithmetic with any group of integers: $Z_n = \{ 0, 1, \dots, n-1 \}$
- form a commutative ring for addition
- with a multiplicative identity
- note some peculiarities
 - if $(a+b) \equiv (a+c) \pmod{n}$ then $b \equiv c \pmod{n}$
 - but $(ab) \equiv (ac) \pmod{n}$ then $b \equiv c \pmod{n}$ only if a is relatively prime to n

Modulo 8 Example

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

(a) Addition modulo 8

Greatest Common Divisor (GCD)

- a common problem in number theory
- GCD (a,b) of a and b is the largest number that divides evenly into both a and b
 - eg $\text{GCD}(60,24) = 12$
- often want **no common factors** (except 1) and hence numbers are **relatively prime**
 - eg $\text{GCD}(8,15) = 1$
 - hence 8 & 15 are relatively prime

Euclid's GCD Algorithm

- an efficient way to find the $\text{GCD}(a,b)$
- uses theorem that:
 - $\text{GCD}(a, b) = \text{GCD}(b, a \bmod b)$
- **Euclid's Algorithm** to compute $\text{GCD}(a,b)$:
 - $A=a$, $B=b$
 - while $B>0$
 - $R = A \bmod B$
 - $A = B$, $B = R$
 - return A

Example GCD(1970,1066)

1970 = 1 × 1066 + 904	gcd(1066, 904)
1066 = 1 × 904 + 162	gcd(904, 162)
904 = 5 × 162 + 94	gcd(162, 94)
162 = 1 × 94 + 68	gcd(94, 68)
94 = 1 × 68 + 26	gcd(68, 26)
68 = 2 × 26 + 16	gcd(26, 16)
26 = 1 × 16 + 10	gcd(16, 10)
16 = 1 × 10 + 6	gcd(10, 6)
10 = 1 × 6 + 4	gcd(6, 4)
6 = 1 × 4 + 2	gcd(4, 2)
4 = 2 × 2 + 0	gcd(2, 0)

Galois Fields

- finite fields play a key role in cryptography
- can show number of elements in a finite field **must** be a power of a prime p^n
- known as Galois fields
- denoted $GF(p^n)$
- in particular often use the fields:
 - $GF(p)$
 - $GF(2^n)$

Galois Fields GF(p)

- GF(p) is the set of integers $\{0, 1, \dots, p-1\}$ with arithmetic operations modulo prime p
- these form a finite field
 - since have multiplicative inverses
- hence arithmetic is “well-behaved” and can do addition, subtraction, multiplication, and division without leaving the field GF(p)

Example GF(7)

\times	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

(b) Multiplication modulo 7

Finding Inverses

- can extend Euclid's algorithm:

EXTENDED EUCLID(m, b)

1. $(A_1, A_2, A_3) = (1, 0, m)$;
 $(B_1, B_2, B_3) = (0, 1, b)$

2. **if** $B_3 = 0$

return $A_3 = \gcd(m, b)$; no inverse

3. **if** $B_3 = 1$

return $B_3 = \gcd(m, b)$; $B_2 = b^{-1} \bmod m$

4. $Q = A_3 \text{ div } B_3$

5. $(T_1, T_2, T_3) = (A_1 - Q B_1, A_2 - Q B_2, A_3 - Q B_3)$

6. $(A_1, A_2, A_3) = (B_1, B_2, B_3)$

7. $(B_1, B_2, B_3) = (T_1, T_2, T_3)$

8. **goto** 2

Inverse of 550 in GF(1759)

Q	A1	A2	A3	B1	B2	B3
-	1	0	1759	0	1	550
3	0	1	550	1	-3	109
5	1	-3	109	-5	16	5
21	-5	16	5	106	-339	4
1	106	-339	4	-111	355	1

Polynomial Arithmetic

- can compute using polynomials

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = \sum_{i=0}^n a_i x^i$$

- several alternatives available
 - ordinary polynomial arithmetic
 - poly arithmetic with coords mod p
 - poly arithmetic with coords mod p and polynomials mod M(x)

Ordinary Polynomial Arithmetic

- add or subtract corresponding coefficients
- multiply all terms by each other
- eg

– let $f(x) = x^3 + x^2 + 2$ and $g(x) = x^2 - x + 1$

$$f(x) + g(x) = x^3 + 2x^2 - x + 3$$

$$f(x) - g(x) = x^3 + x + 1$$

$$f(x) \times g(x) = x^5 + 3x^2 - 2x + 2$$

Polynomial Arithmetic with Modulo Coefficients

- when computing value of each coefficient do calculation modulo some value
- could be modulo any prime
- but we are most interested in mod 2
 - ie all coefficients are 0 or 1
 - eg. let $f(x) = x^3 + x^2$ and $g(x) = x^2 + x + 1$
 $f(x) + g(x) = x^3 + x + 1$
 $f(x) \times g(x) = x^5 + x^2$

Modular Polynomial Arithmetic

- can write any polynomial in the form:
 - $f(x) = q(x) g(x) + r(x)$
 - can interpret $r(x)$ as being a remainder
 - $r(x) = f(x) \bmod g(x)$
- if have no remainder say $g(x)$ divides $f(x)$
- if $g(x)$ has no divisors other than itself & 1 say it is **irreducible** (or prime) polynomial
- arithmetic modulo an irreducible polynomial forms a field

Polynomial GCD

- can find greatest common divisor for polys
 - $c(x) = \text{GCD}(a(x), b(x))$ if $c(x)$ is the poly of greatest degree which divides both $a(x), b(x)$
 - can adapt Euclid's Algorithm to find it:
 - $\text{EUCLID}[a(x), b(x)]$
 - 1. $A(x) = a(x); B(x) = b(x)$
 - 2. **if** $B(x) = 0$ **return** $A(x) = \text{gcd}[a(x), b(x)]$
 - 3. $R(x) = A(x) \text{ mod } B(x)$
 - 4. $A(x) \leftarrow B(x)$
 - 5. $B(x) \leftarrow R(x)$
 - 6. **goto** 2

Modular Polynomial Arithmetic

- can compute in field $\text{GF}(2^n)$
 - polynomials with coefficients modulo 2
 - whose degree is less than n
 - hence must reduce modulo an irreducible poly of degree n (for multiplication only)
- form a finite field
- can always find an inverse
 - can extend Euclid's Inverse algorithm to find

Example GF(2^3)

Table 4.6 Polynomial Arithmetic Modulo ($x^3 + x + 1$)

		000	001	010	011	100	101	110	111
+		0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
000	0	0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
001	1	1	0	$x + 1$	x	$x^2 + 1$	x^2	$x^2 + x + 1$	$x^2 + x$
010	x	x	$x + 1$	0	1	$x^2 + x$	$x^2 + x + 1$	x^2	$x^2 + 1$
011	$x + 1$	$x + 1$	x	1	0	$x^2 + x + 1$	$x^2 + x$	$x^2 + 1$	x^2
100	x^2	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$	0	1	x	$x + 1$
101	$x^2 + 1$	$x^2 + 1$	x^2	$x^2 + x + 1$	$x^2 + x$	1	0	$x + 1$	x
110	$x^2 + x$	$x^2 + x + 1$	x^2	$x^2 + 1$	x	$x + 1$	0	1	
111	$x^2 + x + 1$	$x^2 + x + 1$	$x^2 + 1$	x^2	$x + 1$	x	1	0	

(a) Addition

		000	001	010	011	100	101	110	111
x		0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
000	0	0	0	0	0	0	0	0	0
001	1	0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
010	x	0	x	x^2	$x^2 + x$	$x + 1$	1	$x^2 + x + 1$	$x^2 + 1$
011	$x + 1$	0	$x + 1$	$x^2 + x$	$x^2 + 1$	$x^2 + x + 1$	x^2	1	x
100	x^2	0	x^2	$x + 1$	$x^2 + x + 1$	$x^2 + x$	x	$x^2 + 1$	1
101	$x^2 + 1$	0	$x^2 + 1$	1	x^2	x	$x^2 + x + 1$	$x + 1$	$x^2 + x$
110	$x^2 + x$	0	$x^2 + x$	$x^2 + x + 1$	1	$x^2 + 1$	$x + 1$	x	x^2
111	$x^2 + x + 1$	0	$x^2 + x + 1$	$x^2 + 1$	x	1	$x^2 + x$	x^2	$x + 1$

(b) Multiplication

Computational Considerations

- since coefficients are 0 or 1, can represent any such polynomial as a bit string
- addition becomes XOR of these bit strings
- multiplication is shift & XOR
 - cf long-hand multiplication
- modulo reduction done by repeatedly substituting highest power with remainder of irreducible poly (also shift & XOR)

Summary

- have considered:
 - concept of groups, rings, fields
 - modular arithmetic with integers
 - Euclid's algorithm for GCD
 - finite fields $GF(p)$
 - polynomial arithmetic in general and in $GF(2^n)$

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 5 –Advanced Encryption Standard

"It seems very simple."

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

— Talking to Strange Men, Ruth Rendell

Origins

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

AES Requirements

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

AES Evaluation Criteria

- initial criteria:
 - security – effort to practically cryptanalyse
 - cost – computational
 - algorithm & implementation characteristics
- final criteria
 - general security
 - software & hardware implementation ease
 - implementation attacks
 - flexibility (in en/decrypt, keying, other factors)

AES Shortlist

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

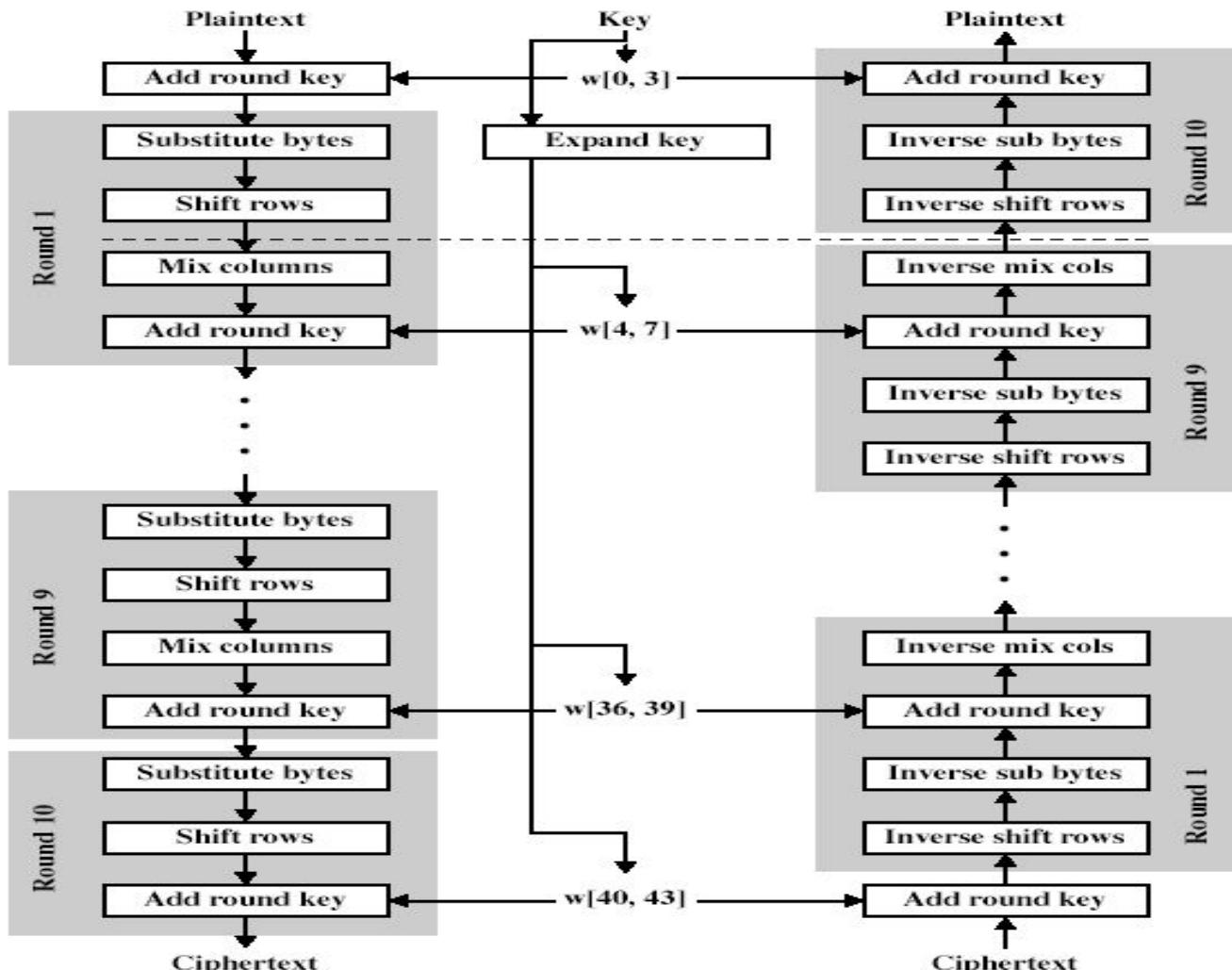
The AES Cipher - Rijndael

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

Rijndael

- processes data as 4 groups of 4 bytes (state)
- 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 multiply of groups)
 - add round key (XOR state with key material)
- initial XOR key material & incomplete last round
- all operations can be combined into XOR and table lookups - hence very fast & efficient

Rijndael



(a) Encryption

(b) Decryption

Byte Substitution

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

Shift Rows

- a circular byte shift in each row
 - 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 does shifts to right
- since state is processed by columns, this step permutes bytes between the columns

Mix Columns

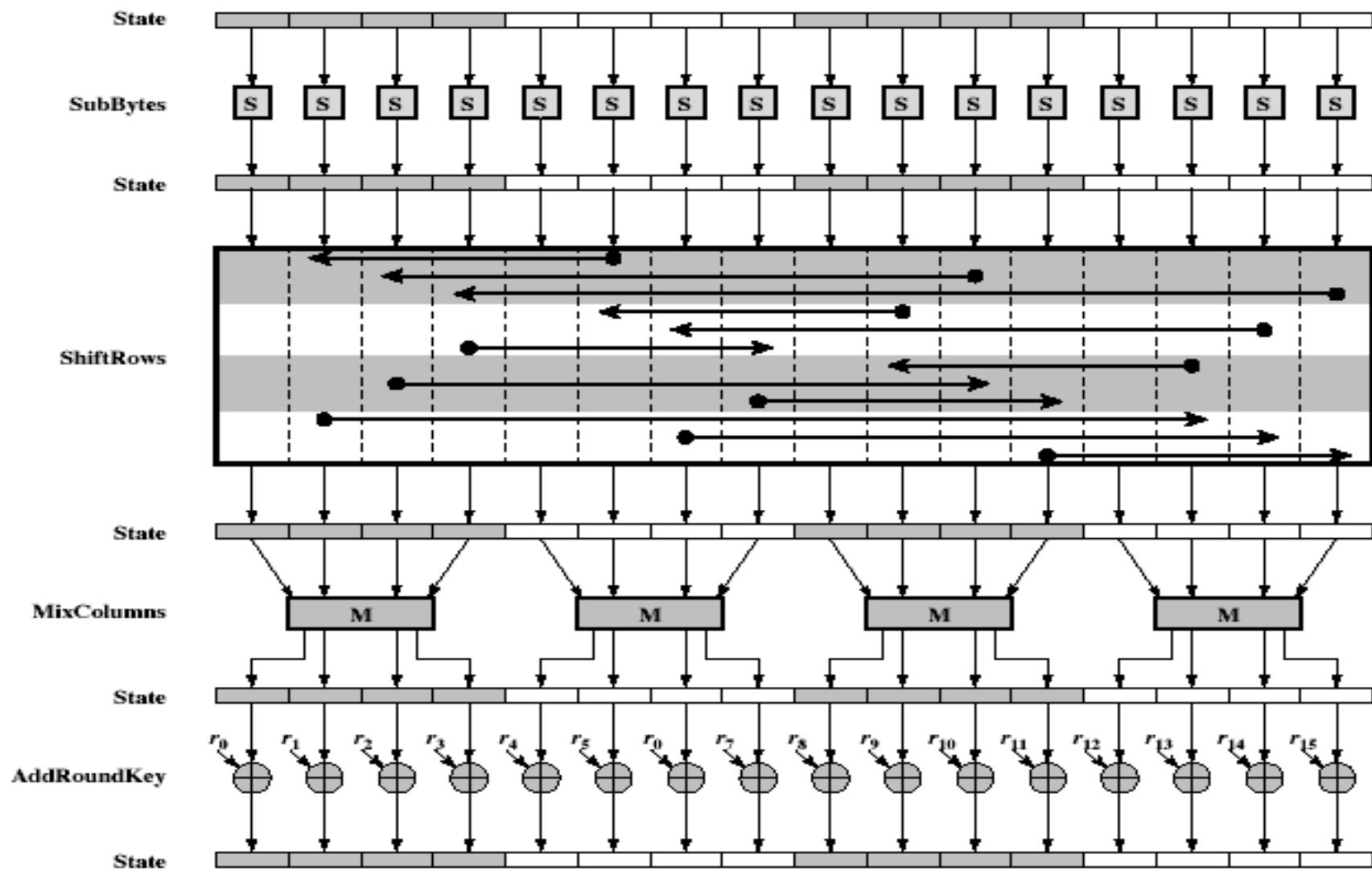
- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column
- effectively a matrix multiplication in $\text{GF}(2^8)$ using prime poly $m(x) = x^8 + x^4 + x^3 + x + 1$

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

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 is identical since XOR is own inverse, just with correct round key
- designed to be as simple as possible

AES Round



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
 - every 4th has S-box + rotate + XOR constant of previous before XOR together
- designed to resist known attacks

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

Implementation Aspects

- can efficiently implement on 8-bit CPU
 - byte substitution works on bytes using a table of 256 entries
 - shift rows is simple byte shifting
 - add round key works on byte XORs
 - mix columns requires matrix multiply in $\text{GF}(2^8)$ which works on byte values, can be simplified to use a table lookup

Implementation Aspects

- can efficiently implement on 32-bit CPU
 - redefine steps to use 32-bit words
 - can precompute 4 tables of 256-words
 - then each column in each round can be computed using 4 table lookups + 4 XORs
 - at a cost of 16Kb to store tables
- designers believe this very efficient implementation was a key factor in its selection as the AES cipher

Summary

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

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 6 – Contemporary Symmetric Ciphers

"I am fairly familiar with all the forms of secret writings, and am myself the author of a trifling monograph upon the subject, in which I analyze one hundred and sixty separate ciphers," said Holmes.

—*The Adventure of the Dancing Men,*
Sir Arthur Conan Doyle

Triple 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

Why Triple-DES?

- why not Double-DES?
 - NOT same as some other single-DES use, but have
- meet-in-the-middle attack
 - works whenever use a cipher twice
 - since $X = E_{K1}[P] = D_{K2}[C]$
 - attack by encrypting P with all keys and store
 - then decrypt C with keys and match X value

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
 - $C = E_{K1} [D_{K2} [E_{K1} [P]]]$
 - 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 there are no practical attacks on two-key Triple-DES, there are some indications
- can use Triple-DES with Three-Keys to avoid even these
 - $C = E_{K3} [D_{K2} [E_{K1} [P]]]$
- has been adopted by some Internet applications, e.g. PGP, S/MIME

Blowfish

- a symmetric block cipher designed by Bruce Schneier in 1993/94
- characteristics
 - fast implementation on 32-bit CPUs
 - compact in use of memory
 - simple structure eases analysis/implementation
 - variable security by varying key size
- has been implemented in various products

Blowfish Key Schedule

- uses a 32 to 448 bit key
- used to generate
 - 18 32-bit subkeys stored in K-array K_j
 - four 8x32 S-boxes stored in $S_{i,j}$
- key schedule consists of:
 - initialize P-array and then 4 S-boxes using π
 - XOR P-array with key bits (reuse as needed)
 - loop repeatedly encrypting data using current P & S and replace successive pairs of P then S values
 - requires 521 encryptions, hence slow in rekeying

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

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

$$L_i = F[R_i] \text{ XOR } R_{i-1};$$

$$L_{17} = R_{16} \text{ XOR } P_{18};$$

$$R_{17} = L_{16} \text{ XOR } i_{17};$$

- where

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

Discussion

- key dependent S-boxes and subkeys, generated using cipher itself, makes analysis very difficult
- changing both halves in each round increases security
- provided key is large enough, brute-force key search is not practical, especially given the high key schedule cost

RC5

- a proprietary cipher owned by RSADSI
- designed by Ronald Rivest (of RSA fame)
- used in various RSADSI products
- can vary key size / data size / no rounds
- very clean and simple design
- easy implementation on various CPUs
- yet still regarded as secure

RC5 Ciphers

- RC5 is a family of ciphers RC5-w/r/b
 - w = word size in bits (16/32/64) nb data=2w
 - r = number of rounds (0..255)
 - b = number of bytes in key (0..255)
- nominal version is RC5-32/12/16
 - ie 32-bit words so encrypts 64-bit data blocks
 - using 12 rounds
 - with 16 bytes (128-bit) secret key

RC5 Key Expansion

- RC5 uses $2r+2$ subkey words (w -bits)
- subkeys are stored in array $S[i]$, $i=0..t-1$
- then the key schedule consists of
 - initializing S to a fixed pseudorandom value, based on constants e and ϕ
 - the byte key is copied (little-endian) into a c -word array L
 - a mixing operation then combines L and S to form the final S array

RC5 Encryption

- split input into two halves A & B

$$L_0 = A + S[0];$$

$$R_0 = B + S[1];$$

for $i = 1$ to r do

$$L_i = ((L_{i-1} \text{ XOR } R_{i-1}) \ll R_{i-1}) + S[2 \times i];$$

$$R_i = ((R_{i-1} \text{ XOR } L_i) \ll L_i) + S[2 \times i + 1];$$

- each round is like 2 DES rounds
- note rotation is main source of non-linearity
- need reasonable number of rounds (eg 12-16)

RC5 Modes

- RFC2040 defines 4 modes used by RC5
 - RC5 Block Cipher, is ECB mode
 - RC5-CBC, is CBC mode
 - RC5-CBC-PAD, is CBC with padding by bytes with value being the number of padding bytes
 - RC5-CTS, a variant of CBC which is the same size as the original message, uses ciphertext stealing to keep size same as original

Block Cipher Characteristics

- features seen in modern block ciphers are:
 - variable key length / block size / no rounds
 - mixed operators, data/key dependent rotation
 - key dependent S-boxes
 - more complex key scheduling
 - operation of full data in each round
 - varying non-linear functions

Stream Ciphers

- process the message bit by bit (as a stream)
- typically have a (pseudo) random **stream key**
- combined (XOR) with plaintext bit by bit
- randomness of **stream key** completely destroys any statistically properties in the message
 - $C_i = M_i \text{ XOR StreamKey}_i$
- what could be simpler!!!!
- but must never reuse stream key
 - otherwise can remove effect and recover messages

Stream Cipher Properties

- some design considerations are:
 - long period with no repetitions
 - statistically random
 - depends on large enough key
 - large linear complexity
 - correlation immunity
 - confusion
 - diffusion
 - use of highly non-linear boolean functions

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
- given a key k of length l bytes

```
for i = 0 to 255 do
    S[i] = i
j = 0
for i = 0 to 255 do
    j = (j + S[i] + k[i mod l]) (mod 256)
    swap (S[i], S[j])
```

RC4 Encryption

- encryption continues shuffling array values
- sum of shuffled pair selects "stream key" value
- tXOR with next byte of message to en/decrypt

i = j = 0

for each message byte M_i

i = (i + 1) (mod 256)

j = (j + S[i]) (mod 256)

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

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

$C_i = M_i \text{ XOR } S[t]$

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

Summary

- have considered:
 - some other modern symmetric block ciphers
 - Triple-DES
 - Blowfish
 - RC5
 - briefly introduced stream ciphers
 - RC4

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 7 – Confidentiality Using Symmetric Encryption

John wrote the letters of the alphabet under the letters in its first lines and tried it against the message.

Immediately he knew that once more he had broken the code. It was extraordinary the feeling of triumph he had. He felt on top of the world. For not only had he done it, had he broken the July code, but he now had the key to every future coded message, since instructions as to the source of the next one must of necessity appear in the current one at the end of each month.

—*Talking to Strange Men*, Ruth Rendell

Confidentiality using Symmetric Encryption

- traditionally symmetric encryption is used to provide message confidentiality
- consider typical scenario
 - workstations on LANs access other workstations & servers on LAN
 - LANs interconnected using switches/routers
 - with external lines or radio/satellite links
- consider attacks and placement in this scenario
 - snooping from another workstation
 - use dial-in to LAN or server to snoop
 - use external router link to enter & snoop
 - monitor and/or modify traffic one external links

Confidentiality using Symmetric 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

Traffic Analysis

- 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

Traffic Analysis

- is monitoring of communications flows between parties
 - useful both in military & commercial spheres
 - can also be used to create a covert channel
- link encryption obscures header details
 - but overall traffic volumes in networks and at end-points is still visible
- traffic padding can further obscure flows
 - but at cost of continuous traffic

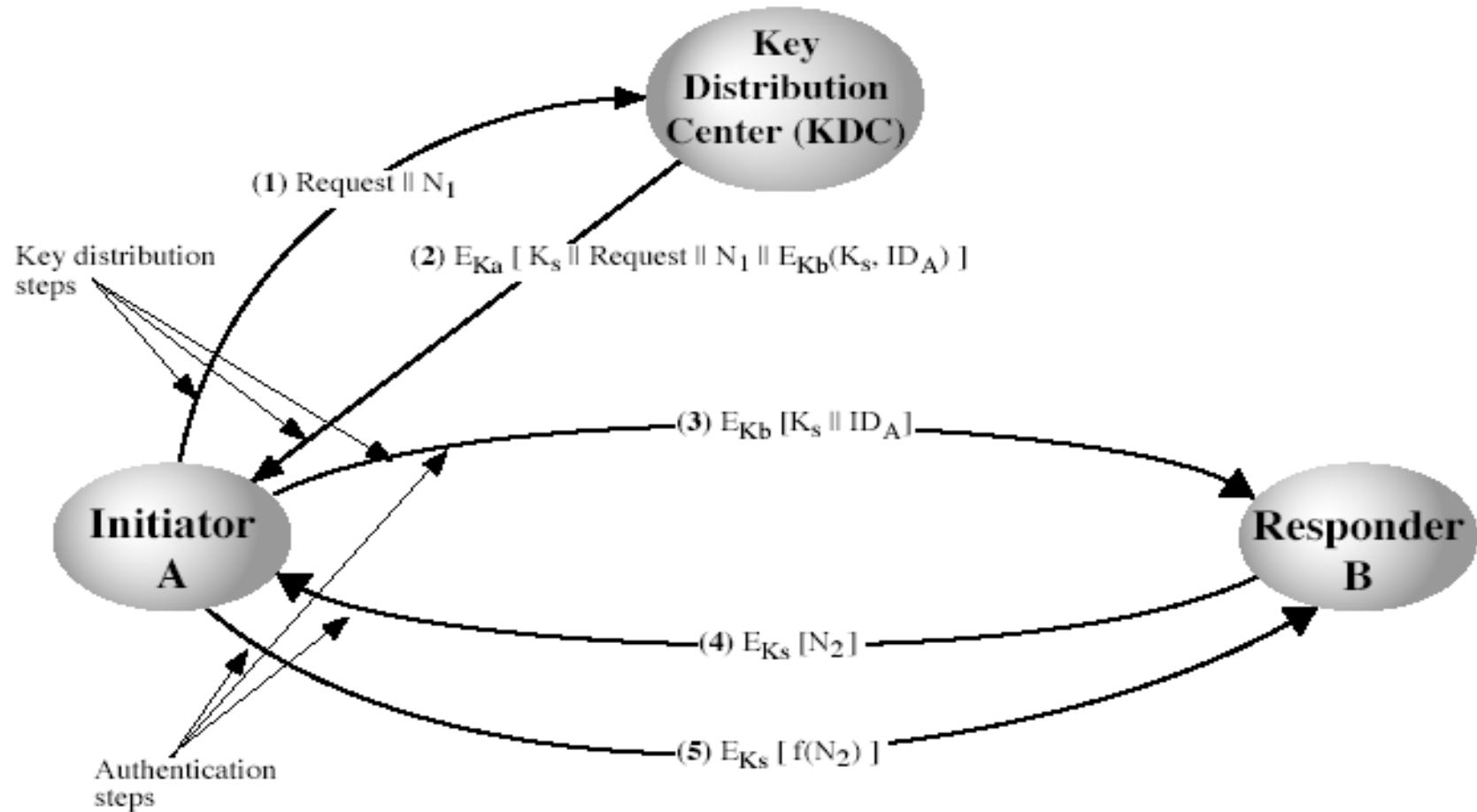
Key Distribution

- symmetric schemes require both parties to share a common secret key
- issue is how to securely distribute this key
- often secure system failure due to a break in the key distribution scheme

Key Distribution

- given parties A and B have various **key distribution** alternatives:
 1. A can select key and physically deliver to B
 2. third party can select & deliver key to A & B
 3. if A & B have communicated previously can use previous key to encrypt a new key
 4. if A & B have secure communications with a third party C, C can relay key between A & B

Key Distribution Scenario



Key Distribution Issues

- hierarchies of KDC's required for large networks, but must trust each other
- session key lifetimes should be limited for greater security
- use of automatic key distribution on behalf of users, but must trust system
- use of decentralized key distribution
- controlling purposes keys are used for

Random Numbers

- many uses of **random numbers** in cryptography
 - nonces in authentication protocols to prevent replay
 - session keys
 - public key generation
 - keystream for a one-time pad
- in all cases its critical that these values be
 - statistically random
 - with uniform distribution, independent
 - unpredictable cannot infer future sequence on previous values

Natural Random Noise

- best source is natural randomness in real world
- find a regular but random event and monitor
- do generally need special h/w to do this
 - eg. radiation counters, radio noise, audio noise, thermal noise in diodes, leaky capacitors, mercury discharge tubes etc
- starting to see such h/w in new CPU's
- problems of **bias** or uneven distribution in signal
 - have to compensate for this when sample and use
 - best to only use a few noisiest bits from each sample

Published Sources

- a few published collections of random numbers
- Rand Co, in 1955, published 1 million numbers
 - generated using an electronic roulette wheel
 - has been used in some cipher designs cf Khafre
- earlier Tippett in 1927 published a collection
- issues are that:
 - these are limited
 - too well-known for most uses

Pseudorandom Number Generators (PRNGs)

- algorithmic technique to create “random numbers”
 - although not truly random
 - can pass many tests of “randomness”

Linear Congruential Generator

- common iterative technique using:

$$X_{n+1} = (aX_n + c) \bmod m$$

- given suitable values of parameters can produce a long random-like sequence
- suitable criteria to have are:
 - function generates a full-period
 - generated sequence should appear random
 - efficient implementation with 32-bit arithmetic
- note that an attacker can reconstruct sequence given a small number of values

Using Block Ciphers as Stream Ciphers

- can use block cipher to generate numbers
- use Counter Mode

$$X_i = E_{Km}[i]$$

- use Output Feedback Mode

$$X_i = E_{Km}[X_{i-1}]$$

- ANSI X9.17 PRNG

- uses date-time + seed inputs and 3 triple-DES encryptions to generate new seed & random

Blum Blum Shub Generator

- based on public key algorithms
- use least significant bit from iterative equation:
 - $x_{i+1} = x_i^2 \bmod n$
 - where $n=p \cdot q$, and primes $p, q \equiv 3 \pmod{4}$
- unpredictable, passes **next-bit** test
- security rests on difficulty of factoring N
- is unpredictable given any run of bits
- slow, since very large numbers must be used
- too slow for cipher use, good for key generation

Summary

- have considered:
 - use of symmetric encryption to protect confidentiality
 - need for good key distribution
 - use of trusted third party KDC's
 - random number generation

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 8 – Introduction to Number Theory

The Devil said to Daniel Webster: "Set me a task I can't carry out, and I'll give you anything in the world you ask for."

Daniel Webster: "Fair enough. Prove that for n greater than 2, the equation $a^n + b^n = c^n$ has no non-trivial solution in the integers."

They agreed on a three-day period for the labor, and the Devil disappeared.

At the end of three days, the Devil presented himself, haggard, jumpy, biting his lip. Daniel Webster said to him, "Well, how did you do at my task? Did you prove the theorem?"

"Eh? No . . . no, I haven't proved it."

"Then I can have whatever I ask for? Money? The Presidency?"

"What? Oh, that—of course. But listen! If we could just prove the following two lemmas—"

—The Mathematical Magpie, Clifton Fadiman

Prime Numbers

- prime numbers only have divisors of 1 and self
 - they cannot be written as a product of other numbers
 - note: 1 is prime, but is generally not of interest
 - eg. 2,3,5,7 are prime, 4,6,8,9,10 are not
 - prime numbers are central to number theory
 - list of prime number less than 200 is:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59
61 67 71 73 79 83 89 97 101 103 107 109 113 127
131 137 139 149 151 157 163 167 173 179 181 191
193 197 199

Prime Factorisation

- to **factor** a number n is to write it as a product of other numbers: $n=a \times b \times c$
- note that factoring a number is relatively hard compared to multiplying the factors together to generate the number
- the **prime factorisation** of a number n is when its written as a product of primes
 - eg. $91=7 \times 13$; $3600=2^4 \times 3^2 \times 5^2$

$$a = \prod_{p \in P} p^{a_p}$$

Relatively Prime Numbers & GCD

- two numbers a , b are **relatively prime** if have **no common divisors** apart from 1
 - eg. 8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor
- conversely can determine the greatest common divisor by comparing their prime factorizations and using least powers
 - eg. $300 = 2^1 \times 3^1 \times 5^2$ $18 = 2^1 \times 3^2$ hence
 $\text{GCD}(18, 300) = 2^1 \times 3^1 \times 5^0 = 6$

Fermat's Theorem

- $a^{p-1} \bmod p = 1$
 - where p is prime and $\gcd(a, p) = 1$
- also known as Fermat's Little Theorem
- useful in public key and primality testing

Euler Totient Function $\phi(n)$

- when doing arithmetic modulo n
- **complete set of residues** is: $0 \dots n-1$
- **reduced set of residues** is those numbers (residues) which are relatively prime to n
 - eg for $n=10$,
 - complete set of residues is $\{0,1,2,3,4,5,6,7,8,9\}$
 - reduced set of residues is $\{1,3,7,9\}$
- number of elements in reduced set of residues is called the **Euler Totient Function $\phi(n)$**

Euler Totient Function $\phi(n)$

- to compute $\phi(n)$ need to count number of elements to be excluded
- in general need prime factorization, but
 - for p (p prime) $\phi(p) = p-1$
 - for $p \cdot q$ (p, q prime) $\phi(p \cdot q) = (p-1)(q-1)$
- eg.
 - $\phi(37) = 36$
 - $\phi(21) = (3-1) \times (7-1) = 2 \times 6 = 12$

Euler's Theorem

- a generalisation of Fermat's Theorem
- $a^{\phi(n)} \text{ mod } N = 1$
 - where $\gcd(a, N) = 1$
- eg.
 - $a=3; n=10; \phi(10)=4;$
 - hence $3^4 = 81 = 1 \text{ mod } 10$
 - $a=2; n=11; \phi(11)=10;$
 - hence $2^{10} = 1024 = 1 \text{ mod } 11$

Primality Testing

- often need to find large prime numbers
- traditionally **sieve** using **trial division**
 - ie. divide by all numbers (primes) in turn less than the square root of the number
 - only works for small numbers
- alternatively can use statistical primality tests based on properties of primes
 - for which all prime numbers satisfy property
 - but some composite numbers, called pseudo-primes, also satisfy the property

Miller Rabin Algorithm

- a test based on Fermat's Theorem
- algorithm is:

TEST (n) is:

1. Find integers k, q , $k > 0$, q odd, so that $(n-1) = 2^k q$
2. Select a random integer a , $1 < a < n-1$
3. **if** $a^q \text{ mod } n = 1$ **then** return ("maybe prime");
4. **for** $j = 0$ **to** $k - 1$ **do**
5. **if** $(a^{2^j q} \text{ mod } n = n-1)$
 then return(" maybe prime ")
6. return ("composite")

Probabilistic Considerations

- if Miller-Rabin returns “composite” the number is definitely not prime
- otherwise is a prime or a pseudo-prime
- chance it detects a pseudo-prime is $< \frac{1}{4}$
- hence if repeat test with different random a then chance n is prime after t tests is:
 - $\Pr(n \text{ prime after } t \text{ tests}) = 1 - 4^{-t}$
 - eg. for $t=10$ this probability is > 0.99999

Prime Distribution

- prime number theorem states that primes occur roughly every $(\ln n)$ integers
- since can immediately ignore evens and multiples of 5, in practice only need test $0.4 \ln(n)$ numbers of size n before locate a prime
 - note this is only the “average” sometimes primes are close together, at other times are quite far apart

Chinese Remainder Theorem

- used to speed up modulo computations
- working modulo a product of numbers
 - eg. $\text{mod } M = m_1 m_2 \dots m_k$
- Chinese Remainder theorem lets us work in each moduli m_i separately
- since computational cost is proportional to size, this is faster than working in the full modulus M

Chinese Remainder Theorem

- can implement CRT in several ways
- to compute $(A \bmod M)$ can firstly compute all $(a_i \bmod m_i)$ separately and then combine results to get answer using:

$$A = \left(\sum_{i=1}^k a_i c_i \right) \bmod M$$

$$c_i = M_i \times (M_i^{-1} \bmod m_i) \quad \text{for } 1 \leq i \leq k$$

Primitive Roots

- from Euler's theorem have $a^{\phi(n)} \bmod n = 1$
- consider $a^m \bmod n = 1$, $\text{GCD}(a, n) = 1$
 - must exist for $m = \phi(n)$ but may be smaller
 - once powers reach m , cycle will repeat
- if smallest is $m = \phi(n)$ then a is called a **primitive root**
- if p is prime, then successive powers of a "generate" the group $\bmod p$
- these are useful but relatively hard to find

Discrete Logarithms or Indices

- the inverse problem to exponentiation is to find the **discrete logarithm** of a number modulo p
- that is to find x where $a^x = b \text{ mod } p$
- written as $x = \log_a b \text{ mod } p$ or $x = \text{ind}_{a,p}(b)$
- if a is a primitive root then always exists,
otherwise may not
 - $x = \log_3 4 \text{ mod } 13$ (x st $3^x = 4 \text{ mod } 13$) has no answer
 - $x = \log_2 3 \text{ mod } 13 = 4$ by trying successive powers
- whilst exponentiation is relatively easy, finding discrete logarithms is generally a **hard** problem

Summary

- have considered:
 - prime numbers
 - Fermat's and Euler's Theorems
 - Primality Testing
 - Chinese Remainder Theorem
 - Discrete Logarithms

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 9 – Public Key Cryptography and RSA

Every Egyptian received two names, which were known respectively as the true name and the good name, or the great name and the little name; and while the good or little name was made public, the true or great name appears to have been carefully concealed.

—***The Golden Bough, Sir James George Frazer***

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

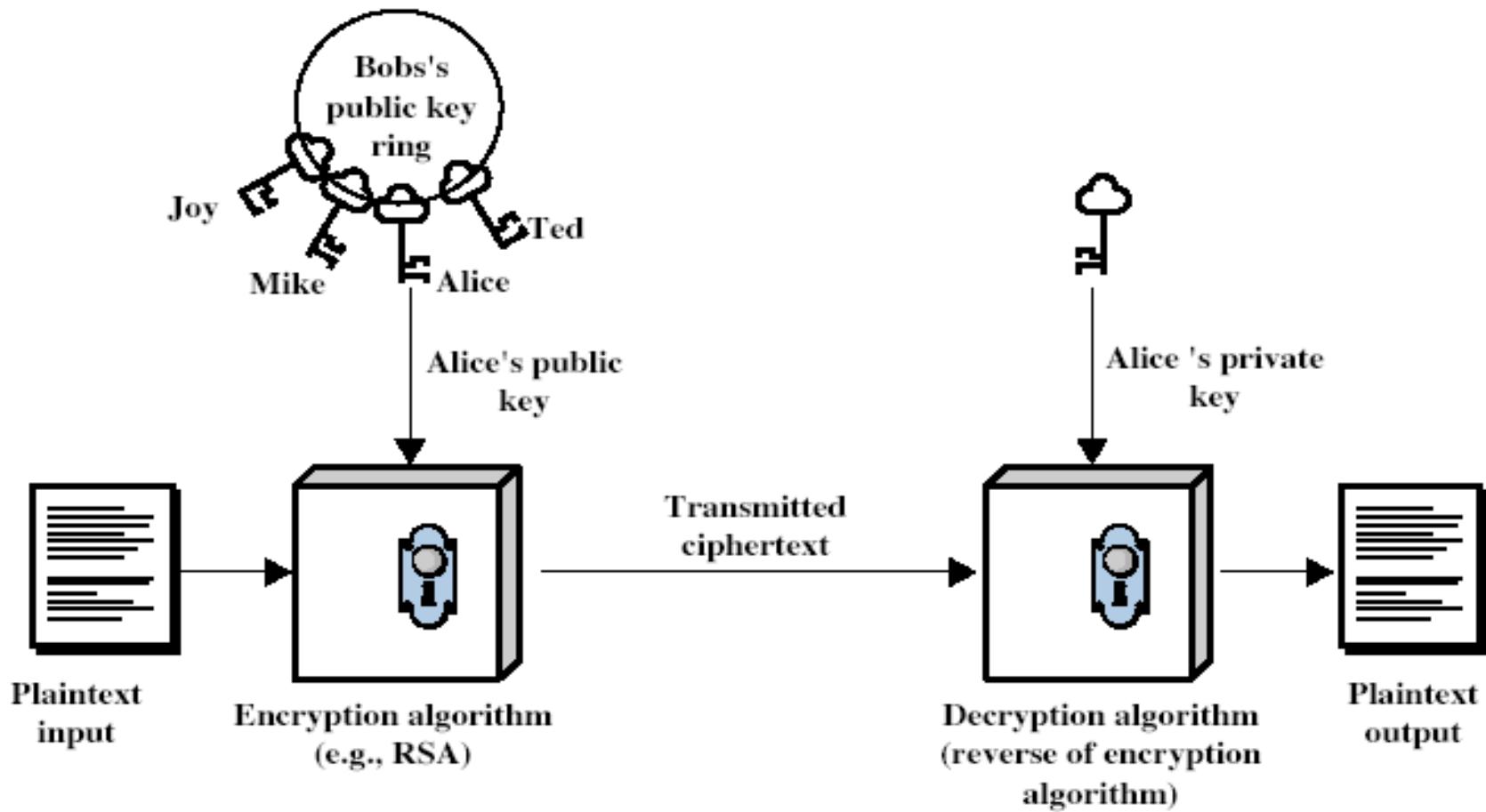
Public-Key Cryptography

- 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

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



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 Characteristics

- Public-Key algorithms rely on two keys with the characteristics that it is:
 - computationally infeasible to find decryption key knowing only algorithm & encryption key
 - 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 (in some schemes)

Public-Key Cryptosystems

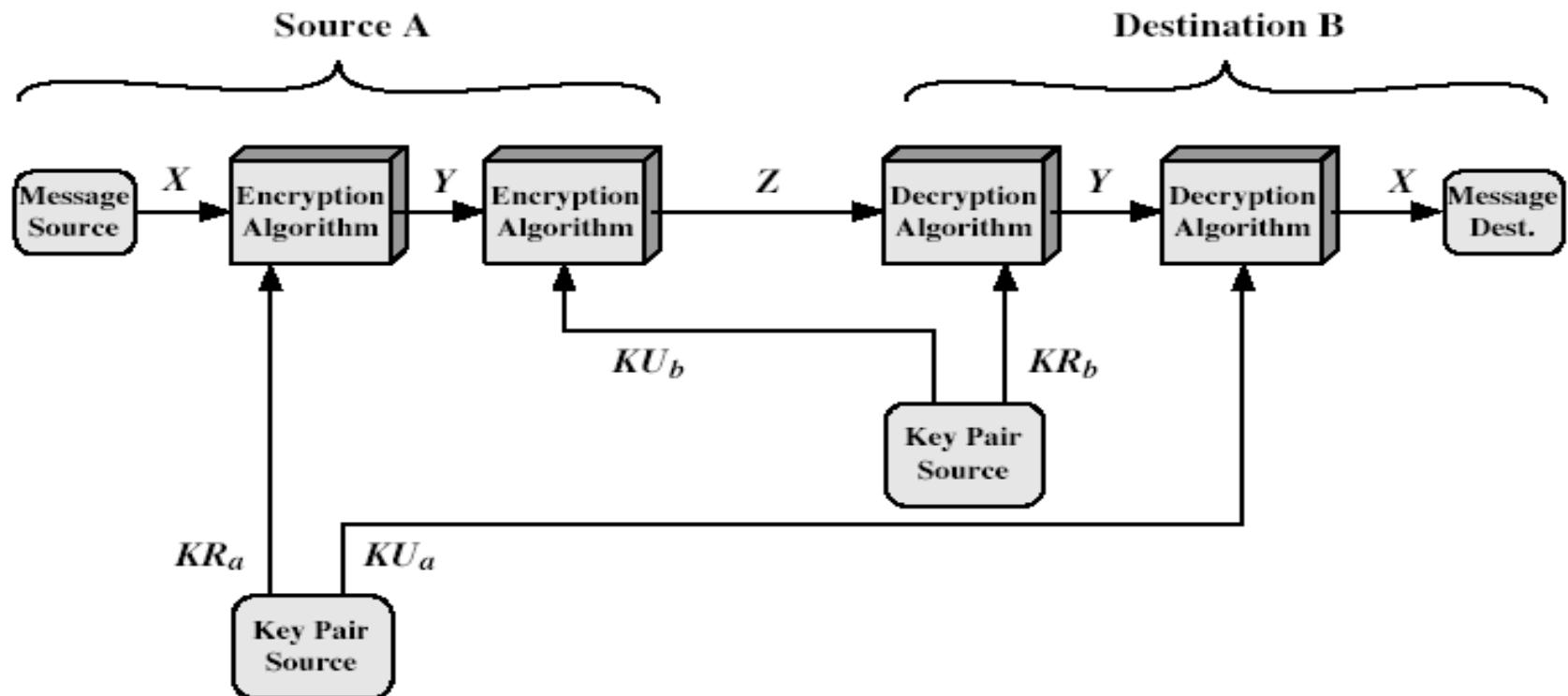


Figure 9.4 Public-Key Cryptosystem: Secrecy and Authentication

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, its just made too hard to do in practise
- 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 $N=p \cdot q$
 - note $\phi(N) = (p-1)(q-1)$
- selecting at random the encryption key e
 - where $1 < e < \phi(N)$, $\gcd(e, \phi(N)) = 1$
- solve following equation to find decryption key d
 - $e \cdot d \equiv 1 \pmod{\phi(N)}$ and $0 \leq d \leq N$
- publish their public encryption key: $KU=\{e,N\}$
- keep secret private decryption key: $KR=\{d,p,q\}$

RSA Use

- to encrypt a message M the sender:
 - obtains **public key** of recipient $KU = \{ e, N \}$
 - computes: $C = M^e \text{ mod } N$, where $0 \leq M < N$
- to decrypt the ciphertext C the owner:
 - uses their private key $KR = \{ d, p, q \}$
 - computes: $M = C^d \text{ mod } N$
- note that the message M must be smaller than the modulus N (block if needed)

Why RSA Works

- because of Euler's Theorem:
- $a^{\phi(n)} \bmod N = 1$
 - where $\gcd(a, N) = 1$
- in RSA have:
 - $N = p \cdot q$
 - $\phi(N) = (p-1)(q-1)$
 - carefully chosen e & d to be inverses $\bmod \phi(N)$
 - hence $e \cdot d = 1 + k \cdot \phi(N)$ for some k
- hence :
$$C^d = (M^e)^d = M^{1+k \cdot \phi(N)} = M^1 \cdot (M^{\phi(N)})^q = M^1 \cdot (1)^q = M^1 = M \bmod N$$

RSA Example

1. Select primes: $p=17 \text{ & } q=11$
2. Compute $n = pq = 17 \times 11 = 187$
3. Compute $\phi(n) = (p-1)(q-1) = 16 \times 10 = 160$
4. Select e : $\gcd(e, 160) = 1$; choose $e=7$
5. Determine d : $de \equiv 1 \pmod{160}$ and $d < 160$
Value is $d=23$ since $23 \times 7 = 161 = 10 \times 160 + 1$
6. Publish public key $KU = \{7, 187\}$
7. Keep secret private key $KR = \{23, 17, 11\}$

RSA Example cont

- sample RSA encryption/decryption is:
- given message $M = 88$ (nb. $88 < 187$)
- encryption:

$$C = 88^7 \bmod 187 = 11$$

- decryption:

$$M = 11^{23} \bmod 187 = 88$$

Exponentiation

- can use the Square and Multiply Algorithm
- a fast, efficient algorithm for exponentiation
- concept is based on repeatedly squaring base
- and multiplying in the ones that are needed to compute the result
- look at binary representation of exponent
- only takes $O(\log_2 n)$ multiples for number n
 - eg. $7^5 = 7^4 \cdot 7^1 = 3 \cdot 7 = 10 \pmod{11}$
 - eg. $3^{129} = 3^{128} \cdot 3^1 = 5 \cdot 3 = 4 \pmod{11}$

Exponentiation

$c \leftarrow 0; d \leftarrow 1$

for $i \leftarrow k$ **downto** 0

do $c \leftarrow 2 \times c$

$d \leftarrow (d \times d) \bmod n$

if $b_i = 1$

then $c \leftarrow c + 1$

$d \leftarrow (d \times a) \bmod n$

return d

RSA Key Generation

- users of RSA must:
 - determine two primes at random - p, q
 - select either e or d and compute the other
- primes p, q must not be easily derived from modulus $N=p \cdot q$
 - means must be sufficiently large
 - typically guess and use probabilistic test
- exponents e, d are inverses, so use Inverse algorithm to compute the other

RSA Security

- three approaches to attacking RSA:
 - brute force key search (infeasible given size of numbers)
 - mathematical attacks (based on difficulty of computing $\phi(N)$, by factoring modulus N)
 - timing attacks (on running of decryption)

Factoring Problem

- mathematical approach takes 3 forms:
 - factor $N=p \cdot q$, hence find $\phi(N)$ and then d
 - determine $\phi(N)$ directly and find d
 - find d directly
- currently believe all equivalent to factoring
 - have seen slow improvements over the years
 - as of Aug-99 best is 130 decimal digits (512) bit with GNFS
 - biggest improvement comes from improved algorithm
 - cf “Quadratic Sieve” to “Generalized Number Field Sieve”
 - barring dramatic breakthrough 1024+ bit RSA secure
 - ensure p, q of similar size and matching other constraints

Timing Attacks

- developed in mid-1990's
- exploit timing variations in operations
 - eg. multiplying by small vs large number
 - or IF's varying which instructions executed
- infer operand size based on time taken
- RSA exploits time taken in exponentiation
- countermeasures
 - use constant exponentiation time
 - add random delays
 - blind values used in calculations

Summary

- have considered:
 - principles of public-key cryptography
 - RSA algorithm, implementation, security

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 10 – Key Management; Other Public Key Cryptosystems

*No Singhalese, whether man or woman,
would venture out of the house without a
bunch of keys in his hand, for without such
a talisman he would fear that some devil
might take advantage of his weak state to
slip into his body.*

—***The Golden Bough, Sir James George
Frazer***

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 Announcement

- users distribute public keys to recipients or broadcast to community at large
 - eg. append PGP keys to email messages or post to news groups or email list
- major weakness is forgery
 - anyone can create a key claiming to be someone else and broadcast it
 - until forgery is discovered can masquerade as claimed user

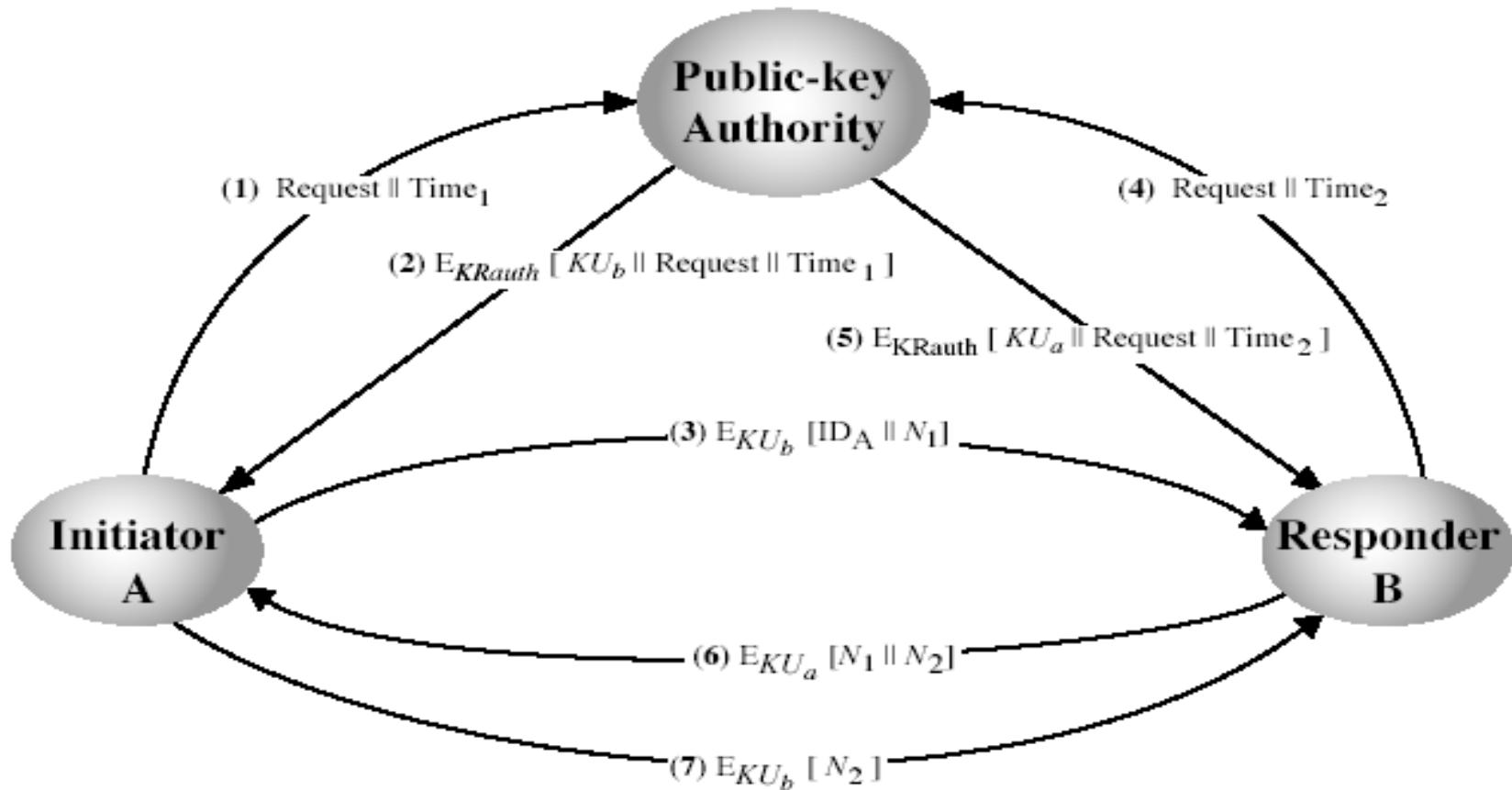
Publicly Available Directory

- can obtain greater security by registering keys with a public directory
- directory must be trusted with properties:
 - contains {name,public-key} entries
 - participants register securely with directory
 - participants can replace key at any time
 - directory is periodically published
 - directory can be accessed electronically
- still vulnerable to tampering or forgery

Public-Key Authority

- improve security by tightening control over distribution of keys from directory
- has properties of directory
- and requires users to know public key for the directory
- then users interact with directory to obtain any desired public key securely
 - does require real-time access to directory when keys are needed

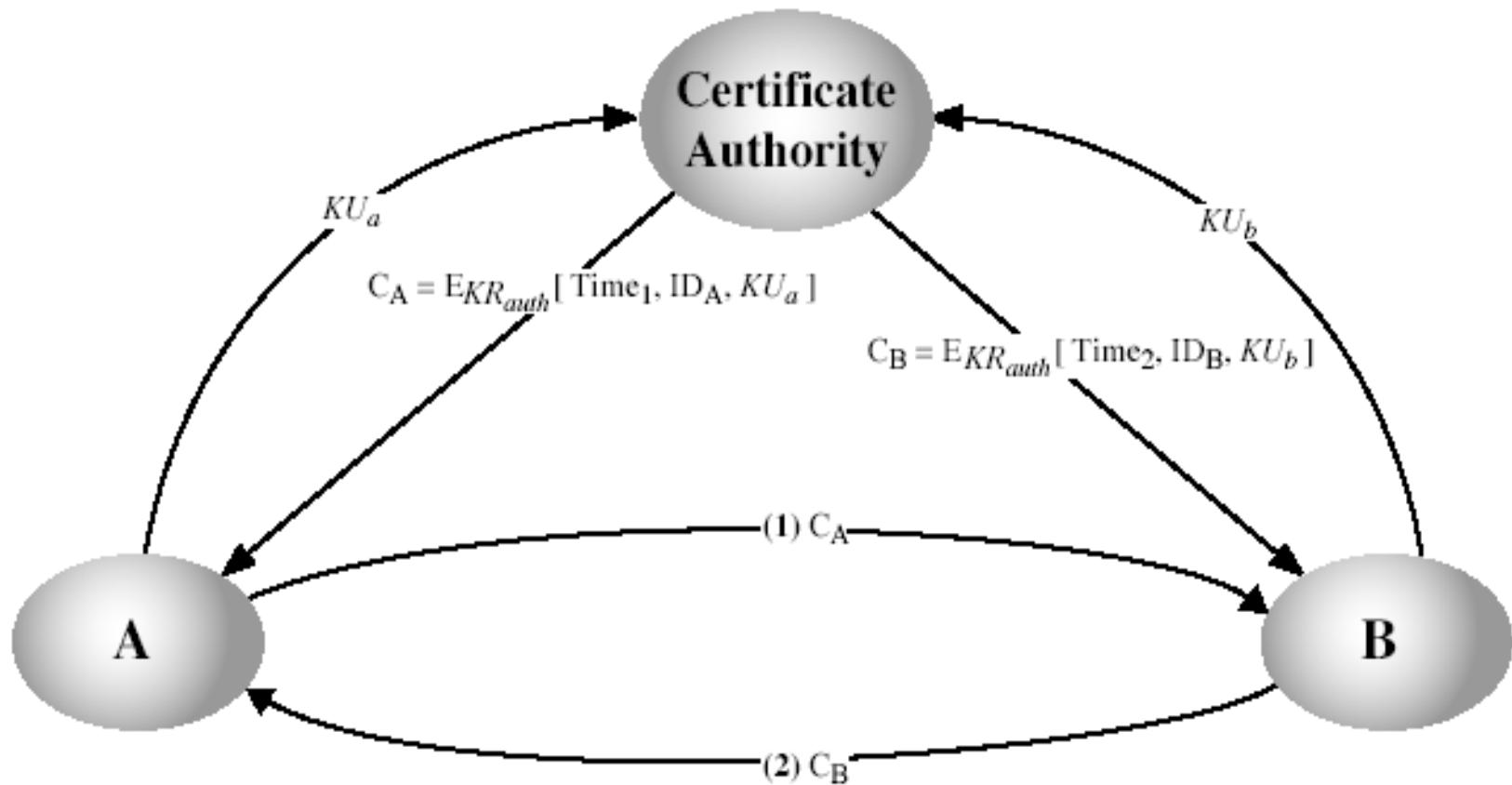
Public-Key Authority



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

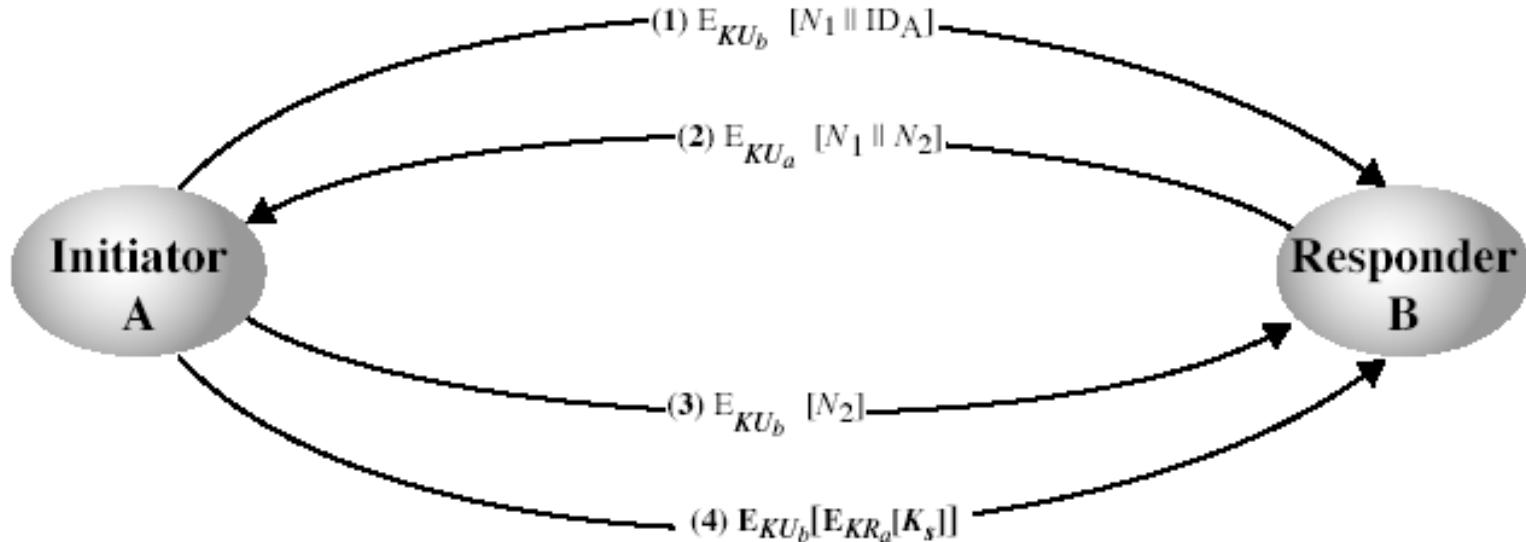
- use previous methods to obtain public-key
- can use for secrecy or authentication
- but public-key algorithms are slow
- so usually want to use private-key encryption to protect message contents
- hence need a session key
- have several alternatives for negotiating a suitable session

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:



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 James Ellis (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
 - α a primitive root mod q
- each user (eg. A) generates their key
 - chooses a secret key (number): $x_A < q$
 - compute their **public key**: $y_A = \alpha^{x_A} \text{ mod } q$
- each user makes public that key y_A

Diffie-Hellman Key Exchange

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

$$K_{AB} = \alpha^{x_A \cdot x_B} \bmod q$$

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

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

- 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 $\alpha=3$
- select random secret keys:
 - A chooses $x_A=97$, B chooses $x_B=233$
- compute public keys:
 - $y_A = 3^{97} \text{ mod } 353 = 40 \quad (\text{Alice})$
 - $y_B = 3^{233} \text{ mod } 353 = 248 \quad (\text{Bob})$
- compute shared session key as:
$$K_{AB} = y_B^{x_A} \text{ mod } 353 = 248^{97} \text{ mod } 353 = 160 \quad (\text{Alice})$$
$$K_{AB} = y_A^{x_B} \text{ mod } 353 = 40^{233} \text{ mod } 353 = 160 \quad (\text{Bob})$$

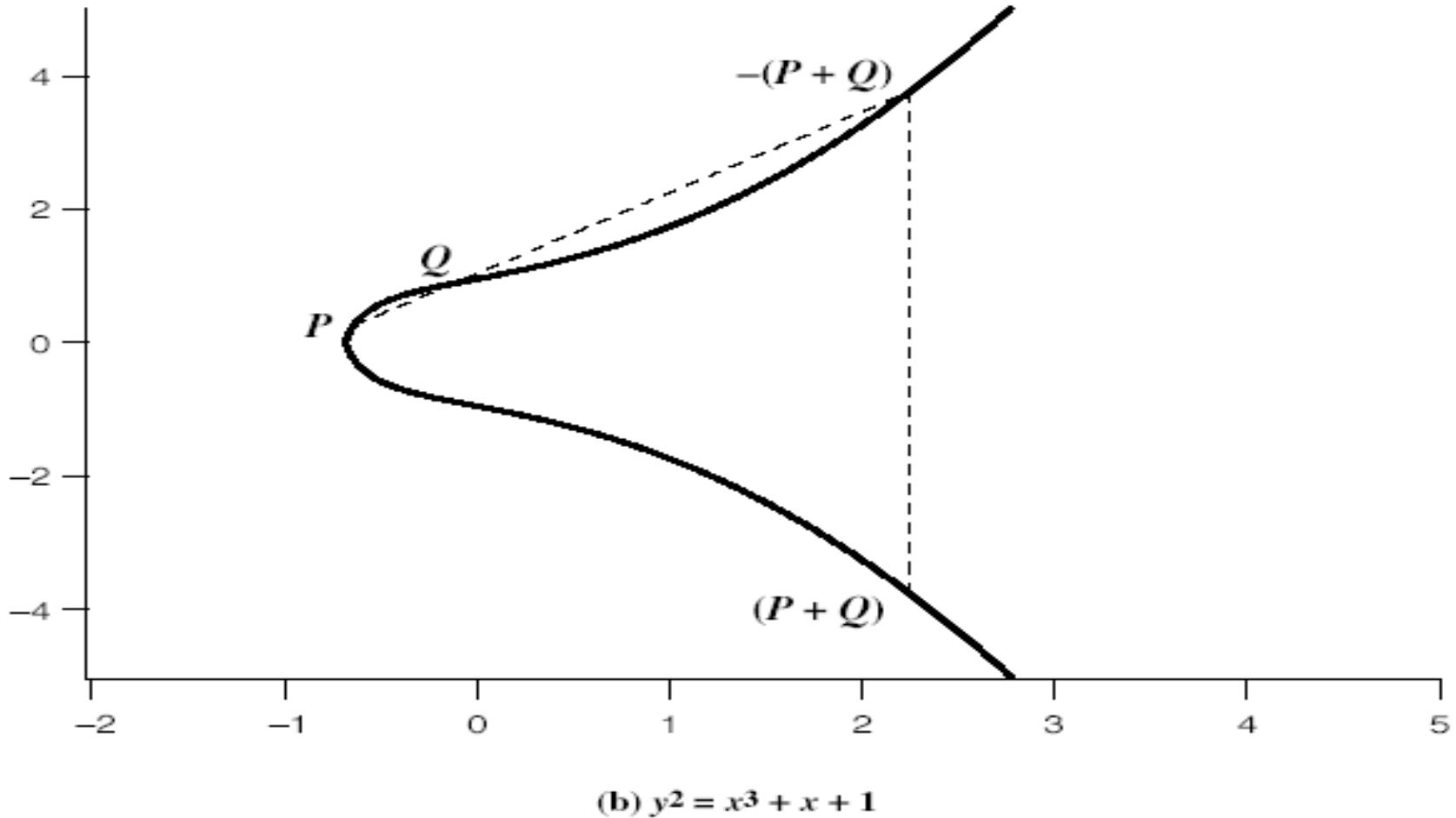
Elliptic Curve Cryptography

- 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

Real Elliptic Curves

- an elliptic curve is defined by an equation in two variables x & y , with coefficients
- consider a cubic elliptic curve of form
 - $y^2 = x^3 + ax + b$
 - where x,y,a,b are all real numbers
 - also define zero point O
- have addition operation for elliptic curve
 - geometrically sum of $Q+R$ is reflection of intersection R

Real Elliptic Curve Example



Finite Elliptic Curves

- Elliptic curve cryptography uses curves whose variables & coefficients are finite
- have two families commonly used:
 - prime curves $E_p(a, b)$ defined over Z_p
 - use integers modulo a prime
 - best in software
 - binary curves $E_{2^m}(a, b)$ defined over $GF(2^n)$
 - use polynomials with binary coefficients
 - best in hardware

Elliptic Curve Cryptography

- ECC addition is analog of modulo multiply
- ECC repeated addition is analog of modulo exponentiation
- need “hard” problem equiv to discrete log
 - $Q = kP$, where Q, P belong to a prime curve
 - is “easy” to compute Q given k, P
 - but “hard” to find k given Q, P
 - known as the elliptic curve logarithm problem
- Certicom example: $E_{23}(9, 17)$

ECC Diffie-Hellman

- can do key exchange analogous to D-H
- users select a suitable curve $E_p(a, b)$
- select base point $G=(x_1, y_1)$ with large order n s.t. $nG=0$
- A & B select private keys $n_A < n, n_B < n$
- compute public keys: $P_A = n_A \times G, P_B = n_B \times G$
- compute shared key: $K = n_A \times P_B, K = n_B \times P_A$
 - same since $K = n_A \times n_B \times G$

ECC Encryption/Decryption

- several alternatives, will consider simplest
- must first encode any message M as a point on the elliptic curve P_m
- select suitable curve & point G as in D-H
- each user chooses private key $n_A < n$
- and computes public key $P_A = n_A \times G$
- to encrypt P_m : $C_m = \{ kG, P_m + kP_b \}$, k random
- decrypt C_m compute:

$$P_m + kP_b - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m$$

ECC Security

- relies on elliptic curve logarithm problem
- fastest method is “Pollard rho method”
- compared to factoring, can use much smaller key sizes than with RSA etc
- for equivalent key lengths computations are roughly equivalent
- hence for similar security ECC offers significant computational advantages

Summary

- have considered:
 - distribution of public keys
 - public-key distribution of secret keys
 - Diffie-Hellman key exchange
 - Elliptic Curve cryptography

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 11 – Message Authentication and Hash Functions

- *At cats' green on the Sunday he took the message from the inside of the pillar and added Peter Moran's name to the two names already printed there in the "Brontosaur" code. The message now read: "Leviathan to Dragon: Martin Hillman, Trevor Allan, Peter Moran: observe and tail." What was the good of it John hardly knew. He felt better, he felt that at last he had made an attack on Peter Moran instead of waiting passively and effecting no retaliation. Besides, what was the use of being in possession of the key to the codes if he never took advantage of it?*
- —**Talking to Strange Men, Ruth Rendell**

Message Authentication

- message authentication is concerned with:
 - protecting the integrity of a message
 - validating identity of originator
 - non-repudiation of origin (dispute resolution)
- will consider the security requirements
- then three alternative functions used:
 - message encryption
 - message authentication code (MAC)
 - hash function

Security Requirements

- disclosure
- traffic analysis
- masquerade
- content modification
- sequence modification
- timing modification
- source repudiation
- destination repudiation

Message Encryption

- message encryption by itself also provides a measure of authentication
- if symmetric encryption is used then:
 - receiver know sender must have created it
 - since only sender and receiver now key used
 - know content cannot of been altered
 - if message has suitable structure, redundancy or a checksum to detect any changes

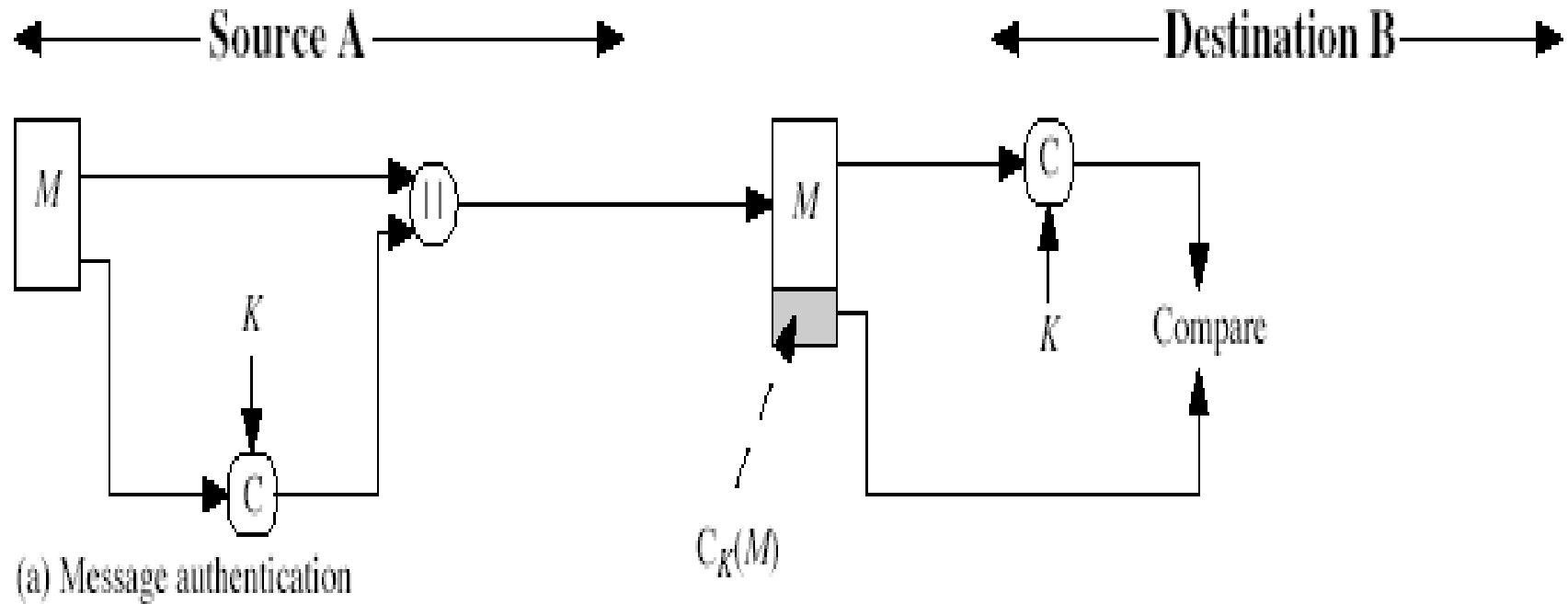
Message Encryption

- if public-key encryption is used:
 - encryption provides no confidence of sender
 - since anyone potentially knows public-key
 - however if
 - sender **signs** message using their private-key
 - then encrypts with recipients public key
 - have both secrecy and authentication
 - again need to recognize corrupted messages
 - but at cost of two public-key uses on message

Message Authentication Code (MAC)

- generated by an algorithm that creates a small fixed-sized block
 - depending on both message and some key
 - like encryption though need not be reversible
- appended to message as a **signature**
- receiver performs same computation on message and checks it matches the MAC
- provides assurance that message is unaltered and comes from sender

Message Authentication Code



Message Authentication Codes

- as shown the MAC provides confidentiality
- 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

$$\text{MAC} = C_K(M)$$

- 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:
 1. knowing a message and MAC, is infeasible to find another message with same MAC
 2. MACs should be uniformly distributed
 3. MAC should depend equally on all bits of the message

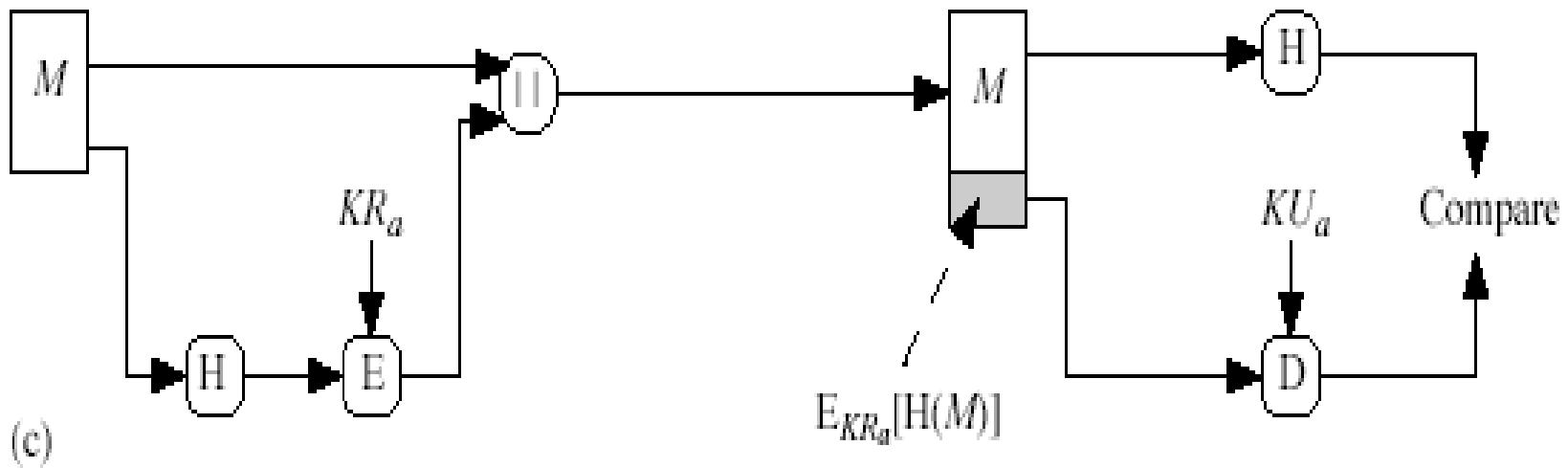
Using Symmetric Ciphers for MACs

- 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 ($16 \leq M \leq 64$) of final block
- but final MAC is now too small for security

Hash Functions

- condenses arbitrary message to fixed size
- 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



Hash Function Properties

- a Hash Function produces a fingerprint of some file/message/data
$$h = H(M)$$
 - condenses a variable-length message M
 - to a fixed-sized fingerprint
- assumed to be public

Requirements for Hash Functions

1. can be applied to any sized message M
2. produces fixed-length output h
3. is easy to compute $h=H(M)$ for any message M
4. given h is infeasible to find x s.t. $H(x)=h$
 - one-way property
5. given x is infeasible to find y s.t. $H(y)=H(x)$
 - weak collision resistance
6. is infeasible to find any x, y s.t. $H(y)=H(x)$
 - strong collision resistance

Simple Hash Functions

- 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)

Birthday Attacks

- might think a 64-bit hash is secure
- but by **Birthday Paradox** is not
- **birthday attack** works thus:
 - opponent generates $2^{m/2}$ variations of a valid message all with essentially the same meaning
 - opponent also generates $2^{m/2}$ variations of a desired fraudulent message
 - two sets of messages are compared to find pair with same hash (probability > 0.5 by birthday paradox)
 - have user sign the valid message, then substitute the forgery which will have a valid signature
- conclusion is that need to use larger MACs

Block Ciphers as Hash Functions

- can use block ciphers as hash functions
 - using $H_0=0$ and zero-pad of final block
 - compute: $H_i = E_{M_i}[H_{i-1}]$
 - and use final block as the hash value
 - similar to CBC but without a key
- resulting hash is too small (64-bit)
 - both due to direct birthday attack
 - and to “meet-in-the-middle” attack
- other variants also susceptible to attack

Hash Functions & MAC Security

- like block ciphers have:
- **brute-force** attacks exploiting
 - strong collision resistance hash have cost $2^{m/2}$
 - have proposal for h/w MD5 cracker
 - 128-bit hash looks vulnerable, 160-bits better
 - MACs with known message-MAC pairs
 - can either attack keyspace (cf key search) or MAC
 - at least 128-bit MAC is needed for security

Hash Functions & MAC Security

- **cryptanalytic attacks** exploit structure
 - like block ciphers want brute-force attacks to be the best alternative
- have a number of analytic attacks on iterated hash functions
 - $CV_i = f[CV_{i-1}, M_i]$; $H(M) = CV_N$
 - typically focus on collisions in function f
 - like block ciphers is often composed of rounds
 - attacks exploit properties of round functions

Summary

- have considered:
 - message authentication using
 - message encryption
 - MACs
 - hash functions
 - general approach & security

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 12 – Hash Algorithms

Each of the messages, like each one he had ever read of Stern's commands, began with a number and ended with a number or row of numbers. No efforts on the part of Mungo or any of his experts had been able to break Stern's code, nor was there any clue as to what the preliminary number and those ultimate numbers signified.

—Talking to Strange Men, Ruth Rendell

Hash Algorithms

- see similarities in the evolution of hash functions & block ciphers
 - increasing power of brute-force attacks
 - leading to evolution in algorithms
 - from DES to AES in block ciphers
 - from MD4 & MD5 to SHA-1 & RIPEMD-160 in hash algorithms
- likewise tend to use common iterative structure as do block ciphers

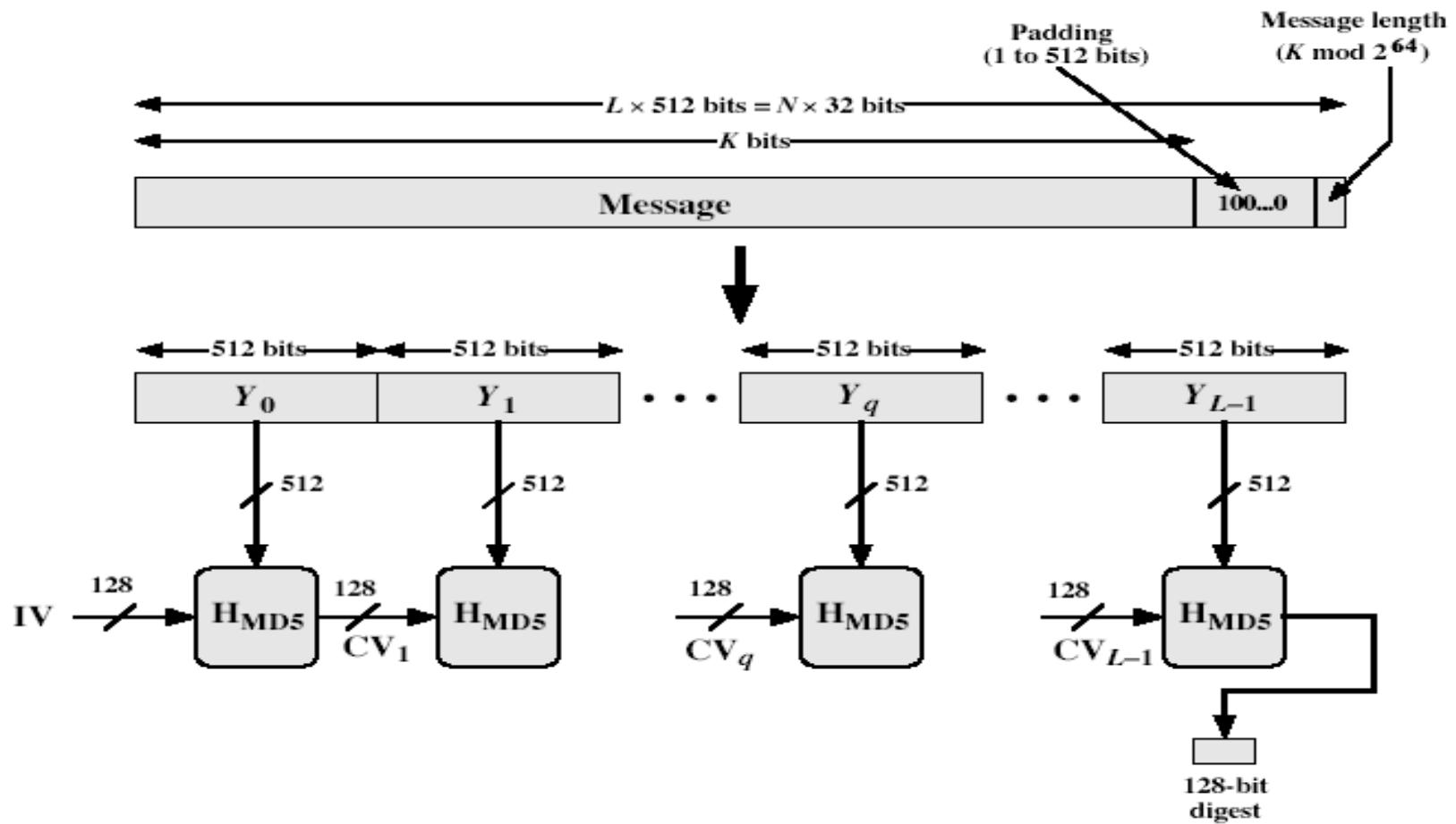
MD5

- designed by Ronald Rivest (the R in RSA)
- latest in a series of MD2, MD4
- produces a 128-bit hash value
- until recently was the most widely used hash algorithm
 - in recent times have both brute-force & cryptanalytic concerns
- specified as Internet standard RFC1321

MD5 Overview

1. pad message so its length is $448 \bmod 512$
2. append a 64-bit length value to message
3. initialise 4-word (128-bit) MD buffer (A,B,C,D)
4. process message in 16-word (512-bit) blocks:
 - using 4 rounds of 16 bit operations on message block & buffer
 - add output to buffer input to form new buffer value
5. output hash value is the final buffer value

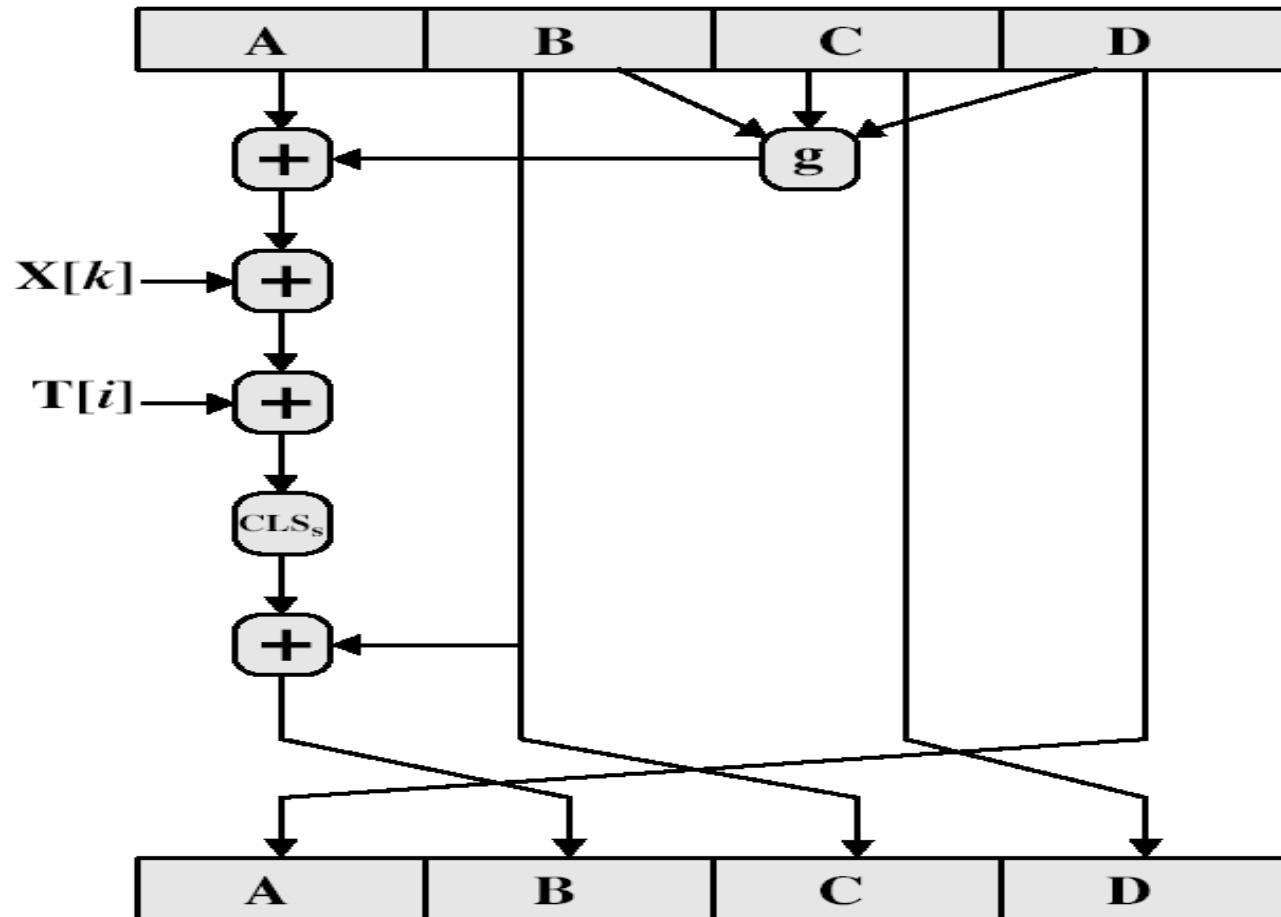
MD5 Overview



MD5 Compression Function

- each round has 16 steps of the form:
$$a = b + ((a+g(b,c,d)+X[k]+T[i]) \ll s)$$
- a,b,c,d refer to the 4 words of the buffer,
but used in varying permutations
 - note this updates 1 word only of the buffer
 - after 16 steps each word is updated 4 times
- where $g(b,c,d)$ is a different nonlinear
function in each round (F,G,H,I)
- $T[i]$ is a constant value derived from sin

MD5 Compression Function



MD4

- precursor to MD5
- also produces a 128-bit hash of message
- has 3 rounds of 16 steps vs 4 in MD5
- design goals:
 - collision resistant (hard to find collisions)
 - direct security (no dependence on "hard" problems)
 - fast, simple, compact
 - favours little-endian systems (eg PCs)

Strength of MD5

- MD5 hash is dependent on all message bits
- Rivest claims security is good as can be
- known attacks are:
 - Berson 92 attacked any 1 round using differential cryptanalysis (but can't extend)
 - Boer & Bosselaers 93 found a pseudo collision (again unable to extend)
 - Dobbertin 96 created collisions on MD compression function (but initial constants prevent exploit)
- conclusion is that MD5 looks vulnerable soon

Secure Hash Algorithm (SHA-1)

- SHA was designed by NIST & NSA in 1993, revised 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
- produces 160-bit hash values
- now the generally preferred hash algorithm
- based on design of MD4 with key differences

SHA Overview

1. pad message so its length is $448 \bmod 512$
2. append a 64-bit length value to message
3. initialise 5-word (160-bit) buffer (A,B,C,D,E) to (67452301,efcdab89,98badcfe,10325476,c3d2e1f0)
4. process message in 16-word (512-bit) chunks:
 - expand 16 words into 80 words by mixing & shifting
 - use 4 rounds of 20 bit operations on message block & buffer
 - add output to input to form new buffer value
5. output hash value is the final buffer value

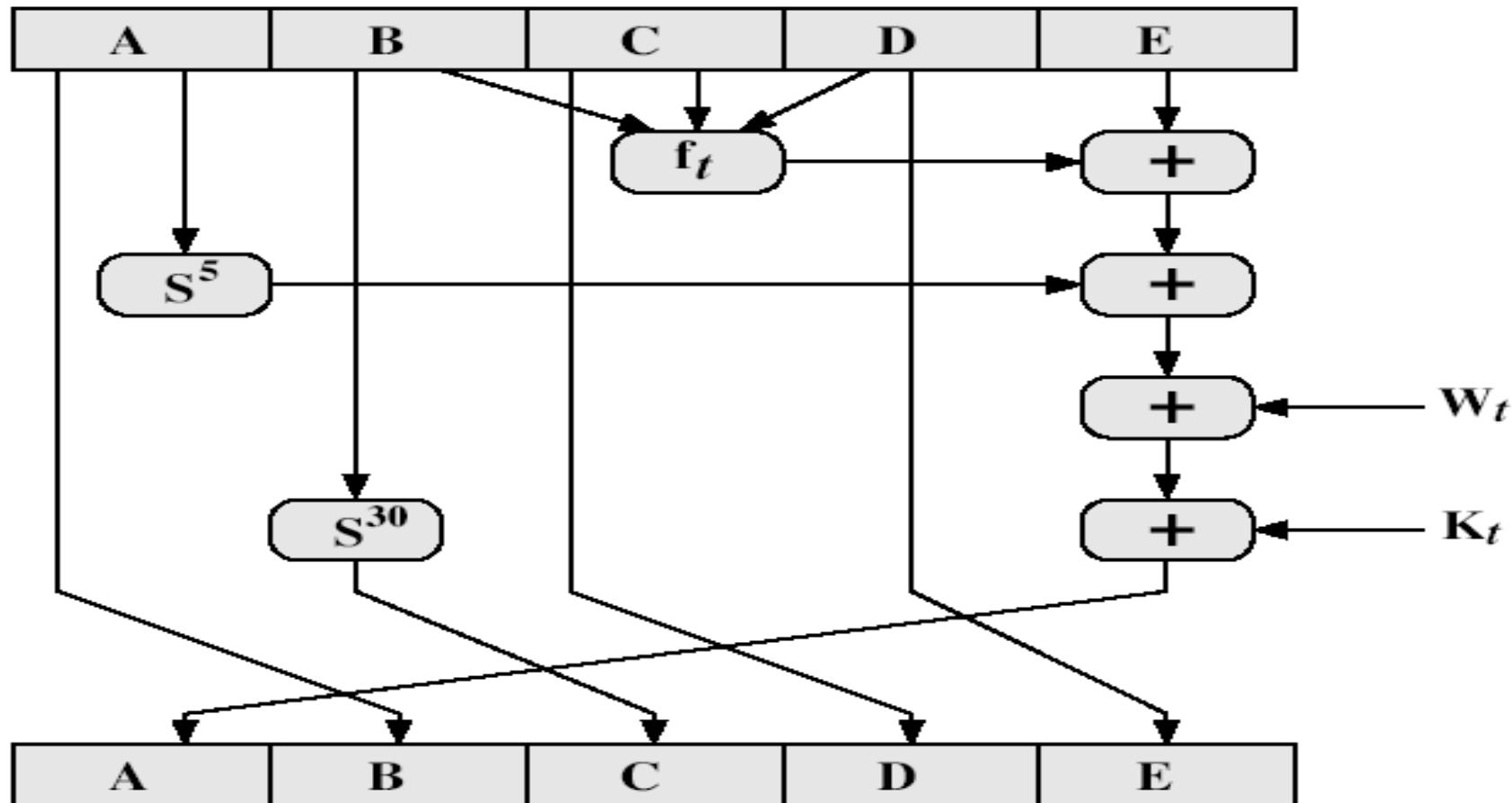
SHA-1 Compression Function

- each round has 20 steps which replaces the 5 buffer words thus:

$$(A, B, C, D, E) \leftarrow (E + f(t, B, C, D) + (A \ll 5) + W_t + K_t), A, (B \ll 30), C, D$$

- a, b, c, d refer to the 4 words of the buffer
- t is the step number
- $f(t, B, C, D)$ is nonlinear function for round
- W_t is derived from the message block
- K_t is a constant value derived from sin

SHA-1 Compression Function



SHA-1 verses MD5

- brute force attack is harder (160 vs 128 bits for MD5)
- not vulnerable to any known attacks (compared to MD4/5)
- a little slower than MD5 (80 vs 64 steps)
- both designed as simple and compact
- optimised for big endian CPU's (vs MD5 which is optimised for little endian CPU's)

Revised Secure Hash Standard

- NIST have issued a revision FIPS 180-2
- adds 3 additional hash algorithms
- 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

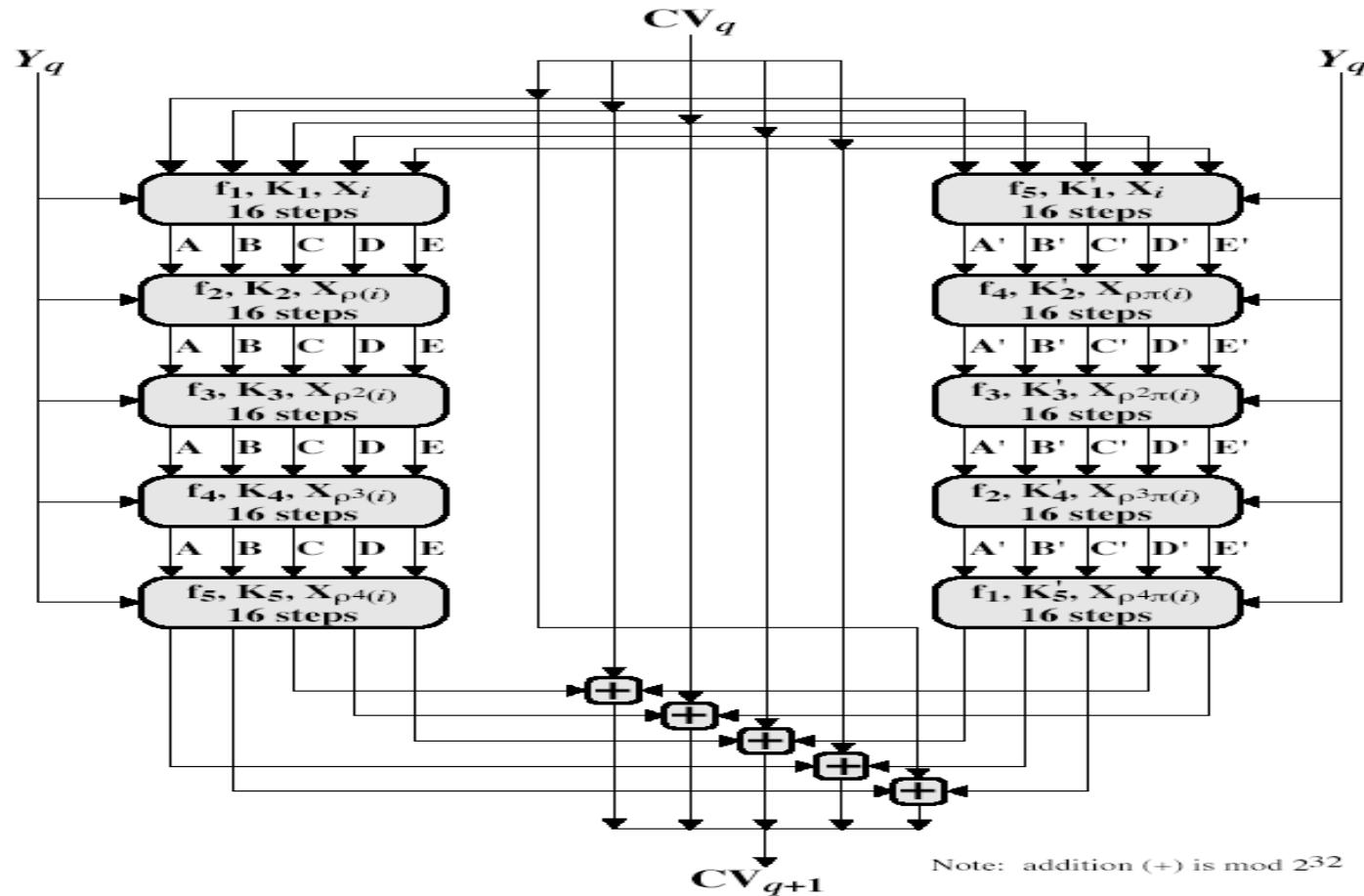
RIPEMD-160

- RIPEMD-160 was developed in Europe as part of RIPE project in 96
- by researchers involved in attacks on MD4/5
- initial proposal strengthen following analysis to become RIPEMD-160
- somewhat similar to MD5/SHA
- uses 2 parallel lines of 5 rounds of 16 steps
- creates a 160-bit hash value
- slower, but probably more secure, than SHA

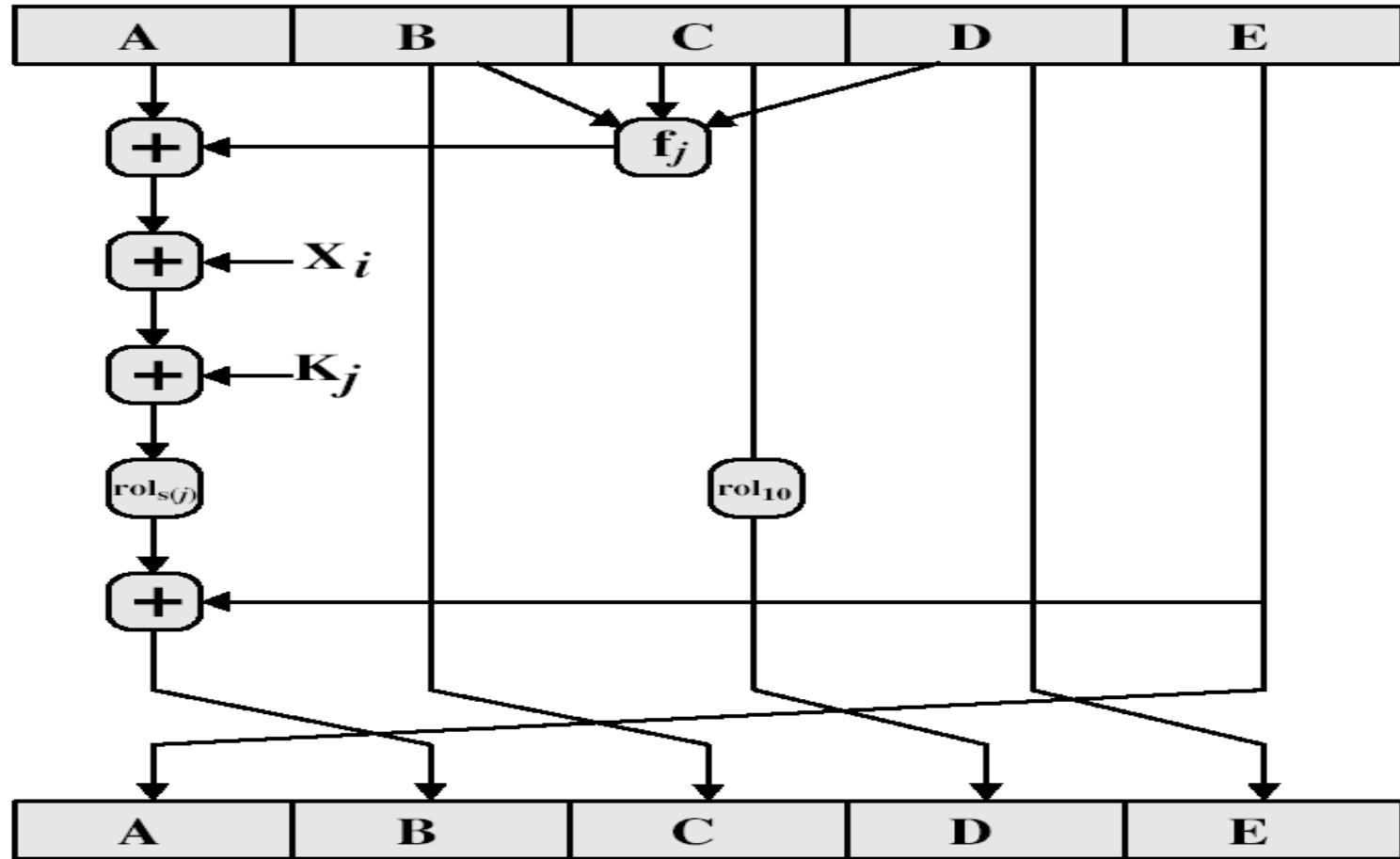
RIPEMD-160 Overview

1. pad message so its length is $448 \bmod 512$
2. append a 64-bit length value to message
3. initialise 5-word (160-bit) buffer (A,B,C,D,E) to (67452301,efcdab89,98badcfe,10325476,c3d2e1f0)
4. process message in 16-word (512-bit) chunks:
 - use 10 rounds of 16 bit operations on message block & buffer – in 2 parallel lines of 5
 - add output to input to form new buffer value
5. output hash value is the final buffer value

RIPEMD-160 Round



RIPEMD-160 Compression Function



RIPEMD-160 Design Criteria

- use 2 parallel lines of 5 rounds for increased complexity
- for simplicity the 2 lines are very similar
- step operation very close to MD5
- permutation varies parts of message used
- circular shifts designed for best results

RIPEMD-160 verses MD5 & SHA-1

- brute force attack harder (160 like SHA-1 vs 128 bits for MD5)
- not vulnerable to known attacks, like SHA-1 though stronger (compared to MD4/5)
- slower than MD5 (more steps)
- all designed as simple and compact
- SHA-1 optimised for big endian CPU's vs RIPEMD-160 & MD5 optimised for little endian CPU's

Keyed Hash Functions as MACs

- have desire to create a MAC using a hash function rather than a block cipher
 - because hash functions are generally faster
 - not limited by export controls unlike block ciphers
- hash includes a key along with the message
- original proposal:
$$\text{KeyedHash} = \text{Hash}(\text{Key} \mid \text{Message})$$
 - some weaknesses were found with this
- eventually led to development of HMAC

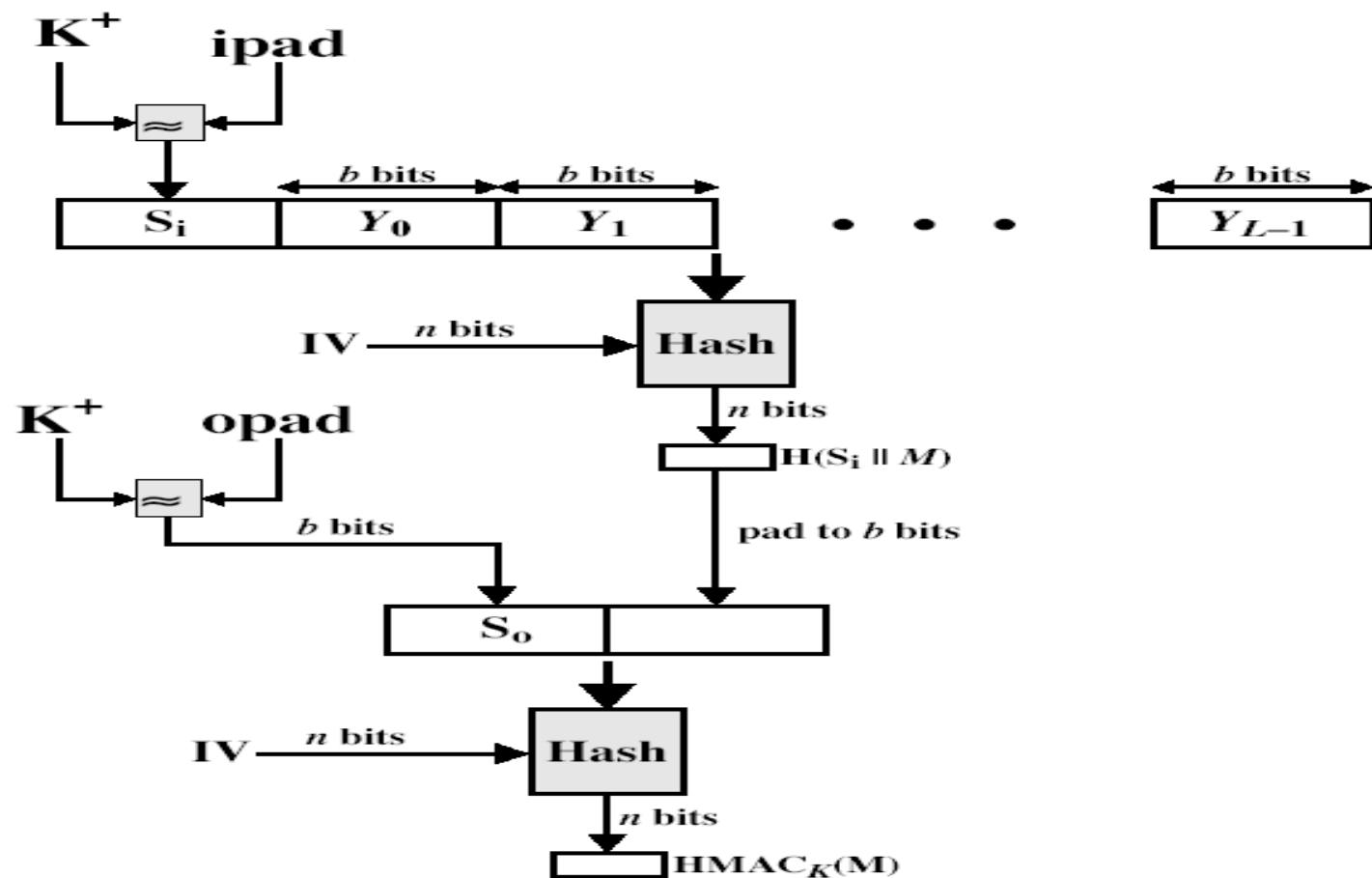
HMAC

- specified as Internet standard RFC2104
- uses hash function on the message:

$$\text{HMAC}_K = \text{Hash} [(K^+ \text{ XOR } \text{opad}) \parallel \\ \text{Hash} [(K^+ \text{ XOR } \text{ipad}) \parallel M]]$$

- 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 of MD5, SHA-1, RIPEMD-160 can be used

HMAC Overview



HMAC Security

- know that the 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 versus security constraints

Summary

- have considered:
 - some current hash algorithms: MD5, SHA-1, RIPEMD-160
 - HMAC authentication using hash function

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 13 –Digital Signatures & Authentication Protocols

To guard against the baneful influence exerted by strangers is therefore an elementary dictate of savage prudence. Hence before strangers are allowed to enter a district, or at least before they are permitted to mingle freely with the inhabitants, certain ceremonies are often performed by the natives of the country for the purpose of disarming the strangers of their magical powers, or of disinfecting, so to speak, the tainted atmosphere by which they are supposed to be surrounded.

—***The Golden Bough, Sir James George Frazer***

Digital Signatures

- have looked at message authentication
 - but does not address issues of lack of trust
- digital signatures provide the ability to:
 - verify author, date & time of signature
 - authenticate message contents
 - be verified by third parties to resolve disputes
- hence include authentication function with additional capabilities

Digital Signature Properties

- must depend on the message signed
- must use information unique to sender
 - to prevent both forgery and denial
- must be relatively easy to produce
- must be relatively easy to recognize & verify
- be computationally infeasible to forge
 - with new message for existing digital signature
 - with fraudulent digital signature for given message
- be practical save digital signature in storage

Direct Digital Signatures

- involve only sender & receiver
- assumed receiver has sender's public-key
- digital signature made by sender signing entire message or hash with private-key
- can encrypt using receivers public-key
- important that sign first then encrypt message & signature
- security depends on sender's private-key

Arbitrated Digital Signatures

- involves use of arbiter A
 - validates any signed message
 - then dated and sent to recipient
- requires suitable level of trust in arbiter
- can be implemented with either private or public-key algorithms
- arbiter may or may not see message

Authentication Protocols

- used to convince parties of each others identity and to exchange session keys
- may be one-way or mutual
- key issues are
 - confidentiality – to protect session keys
 - timeliness – to prevent replay attacks

Replay Attacks

- where a valid signed message is copied and later resent
 - simple replay
 - repetition that can be logged
 - repetition that cannot be detected
 - backward replay without modification
- countermeasures include
 - use of sequence numbers (generally impractical)
 - timestamps (needs synchronized clocks)
 - challenge/response (using unique nonce)

Using Symmetric Encryption

- as discussed previously can use a two-level hierarchy of keys
- usually with a trusted Key Distribution Center (KDC)
 - each party shares own master key with KDC
 - KDC generates session keys used for connections between parties
 - master keys used to distribute these to them

Needham-Schroeder Protocol

- original third-party key distribution protocol
- for session between A B mediated by KDC
- protocol overview is:
 1. A→KDC: $ID_A \parallel ID_B \parallel N_1$
 2. KDC→A: $E_{Ka}[Ks \parallel ID_B \parallel N_1 \parallel E_{Kb}[Ks \parallel ID_A]]$
 3. A→B: $E_{Kb}[Ks \parallel ID_A]$
 4. B→A: $E_{Ks}[N_2]$
 5. A→B: $E_{Ks}[f(N_2)]$

Needham-Schroeder Protocol

- used to securely distribute a new session key for communications between A & B
- but is vulnerable to a replay attack if an old session key has been compromised
 - then message 3 can be resent convincing B that is communicating with A
- modifications to address this require:
 - timestamps (Denning 81)
 - using an extra nonce (Neuman 93)

Using Public-Key Encryption

- have a range of approaches based on the use of public-key encryption
- need to ensure have correct public keys for other parties
- using a central Authentication Server (AS)
- various protocols exist using timestamps or nonces

Denning AS Protocol

- Denning 81 presented the following:
 1. $A \rightarrow AS: ID_A \parallel ID_B$
 2. $AS \rightarrow A: E_{KRas}[ID_A \parallel KU_a \parallel T] \parallel E_{KRas}[ID_B \parallel KU_b \parallel T]$
 3. $A \rightarrow B: E_{KRas}[ID_A \parallel KU_a \parallel T] \parallel E_{KRas}[ID_B \parallel KU_b \parallel T] \parallel E_{KUb}[E_{KRas}[K_s \parallel T]]$
- note session key is chosen by A, hence AS need not be trusted to protect it
- timestamps prevent replay but require synchronized clocks

One-Way Authentication

- required when sender & receiver are not in communications at same time (eg. email)
- have header in clear so can be delivered by email system
- may want contents of body protected & sender authenticated

Using Symmetric Encryption

- can refine use of KDC but can't have final exchange of nonces, vis:
 1. A→KDC: $ID_A \parallel ID_B \parallel N_1$
 2. KDC→A: $E_{Ka}[Ks \parallel ID_B \parallel N_1 \parallel E_{Kb}[Ks||ID_A]]$
 3. A→B: $E_{Kb}[Ks||ID_A] \parallel E_{Ks}[M]$
- does not protect against replays
 - could rely on timestamp in message, though email delays make this problematic

Public-Key Approaches

- have seen some public-key approaches
- if confidentiality is major concern, can use:
 $A \rightarrow B: E_{KUb}[Ks] \parallel E_{Ks}[M]$
 - has encrypted session key, encrypted message
- if authentication needed use a digital signature with a digital certificate:
 $A \rightarrow B: M \parallel E_{KRa}[H(M)] \parallel E_{KRas}[T \parallel ID_A \parallel KU_a]$
 - with message, signature, certificate

Digital Signature Standard (DSS)

- US Govt approved signature scheme FIPS 186
- uses the SHA hash algorithm
- designed by NIST & NSA in early 90's
- DSS is the standard, DSA is the algorithm
- a variant on ElGamal and Schnorr schemes
- creates a 320 bit signature, but with 512-1024 bit security
- security depends on difficulty of computing discrete logarithms

DSA Key Generation

- have shared global public key values (p, q, g):
 - a large prime $p = 2^L$
 - where $L = 512$ to 1024 bits and is a multiple of 64
 - choose q , a 160 bit prime factor of $p-1$
 - choose $g = h^{(p-1)/q}$
 - where $h < p-1$, $h^{(p-1)/q} \pmod{p} > 1$
- users choose private & compute public key:
 - choose $x < q$
 - compute $y = g^x \pmod{p}$

DSA Signature Creation

- to **sign** a message M the 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:
$$r = (g^k \pmod p) \pmod q$$
$$s = (k^{-1} \cdot \text{SHA}(M) + x \cdot r) \pmod q$$
- sends signature (r, s) with message M

DSA Signature Verification

- having received M & signature (r, s)
- to **verify** a signature, recipient computes:

$$w = s^{-1} \pmod{q}$$

$$u1 = (\text{SHA}(M) \cdot w) \pmod{q}$$

$$u2 = (r \cdot w) \pmod{q}$$

$$v = (g^{u1} \cdot y^{u2} \pmod{p}) \pmod{q}$$

- if $v=r$ then signature is verified
- see book web site for details of proof why

Summary

- have considered:
 - digital signatures
 - authentication protocols (mutual & one-way)
 - digital signature standard

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 14 – Authentication Applications

We cannot enter into alliance with neighboring princes until we are acquainted with their designs.

—*The Art of War, Sun Tzu*

Authentication Applications

- will consider authentication functions
- developed to support application-level authentication & digital signatures
- will consider Kerberos – a private-key authentication service
- then X.509 directory authentication service

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

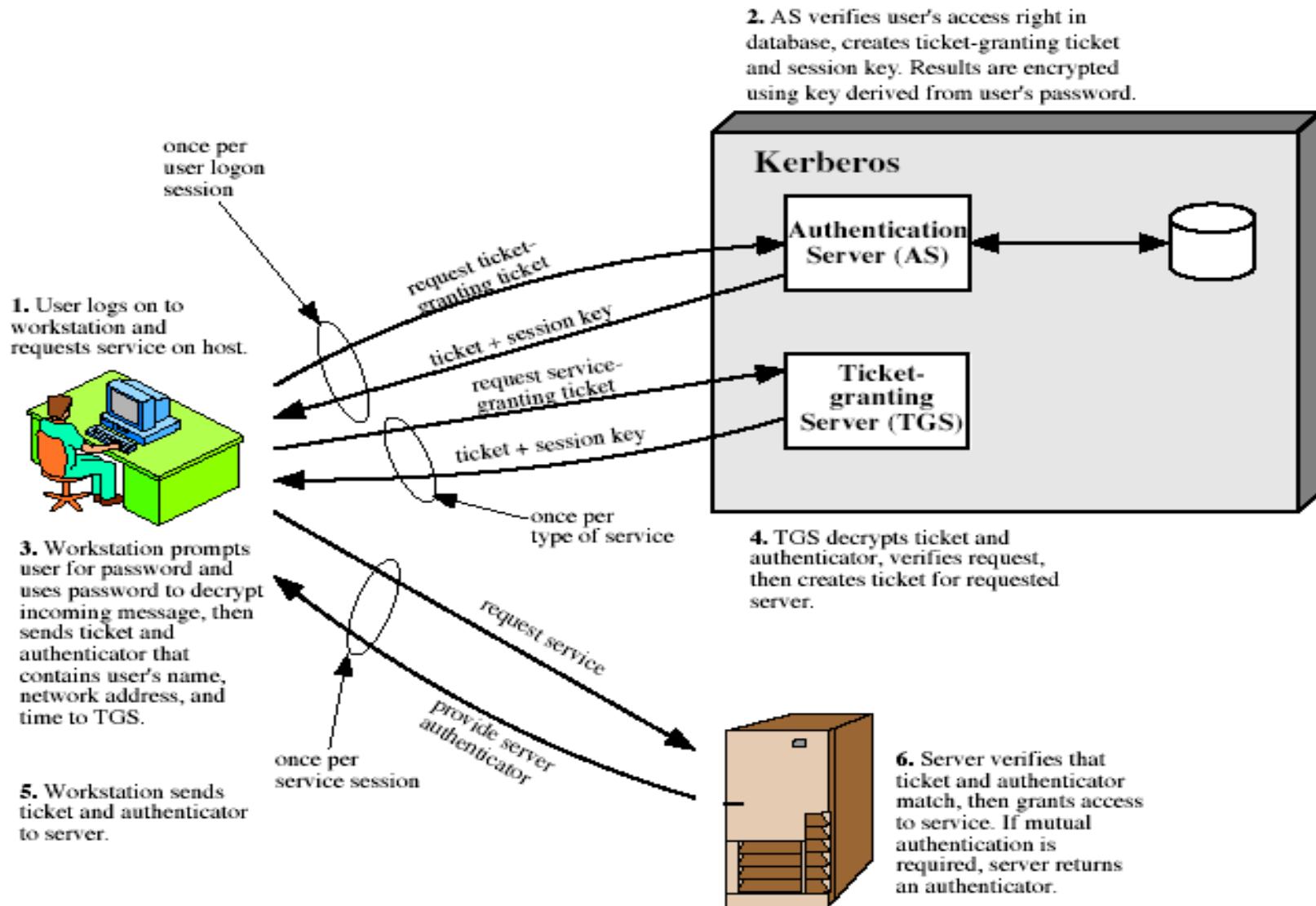
Kerberos Requirements

- first published report identified its requirements as:
 - security
 - reliability
 - transparency
 - scalability
- implemented using an authentication protocol based on Needham-Schroeder

Kerberos 4 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 (ticket granting ticket TGT)
- have a Ticket Granting server (TGS)
 - users subsequently request access to other services from TGS on basis of users TGT

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 Version 5

- developed in mid 1990's
- 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
- specified as Internet standard RFC 1510

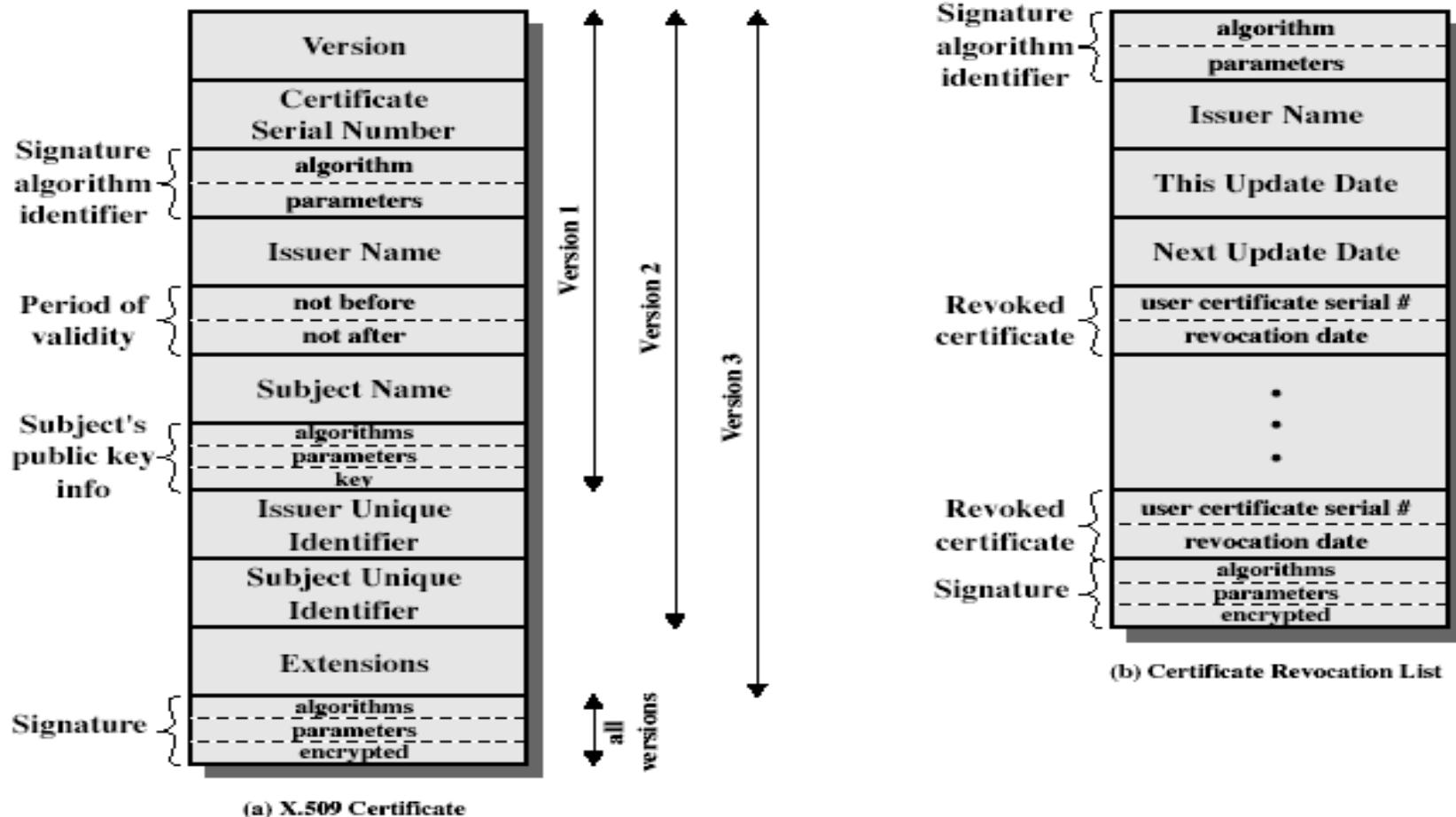
X.509 Authentication Service

- part of CCITT X.500 directory service standards
 - distributed servers maintaining some info database
- defines framework for authentication services
 - directory may store public-key certificates
 - with public key of user
 - signed by certification authority
- also defines authentication protocols
- uses public-key crypto & digital signatures
 - algorithms not standardised, but RSA recommended

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



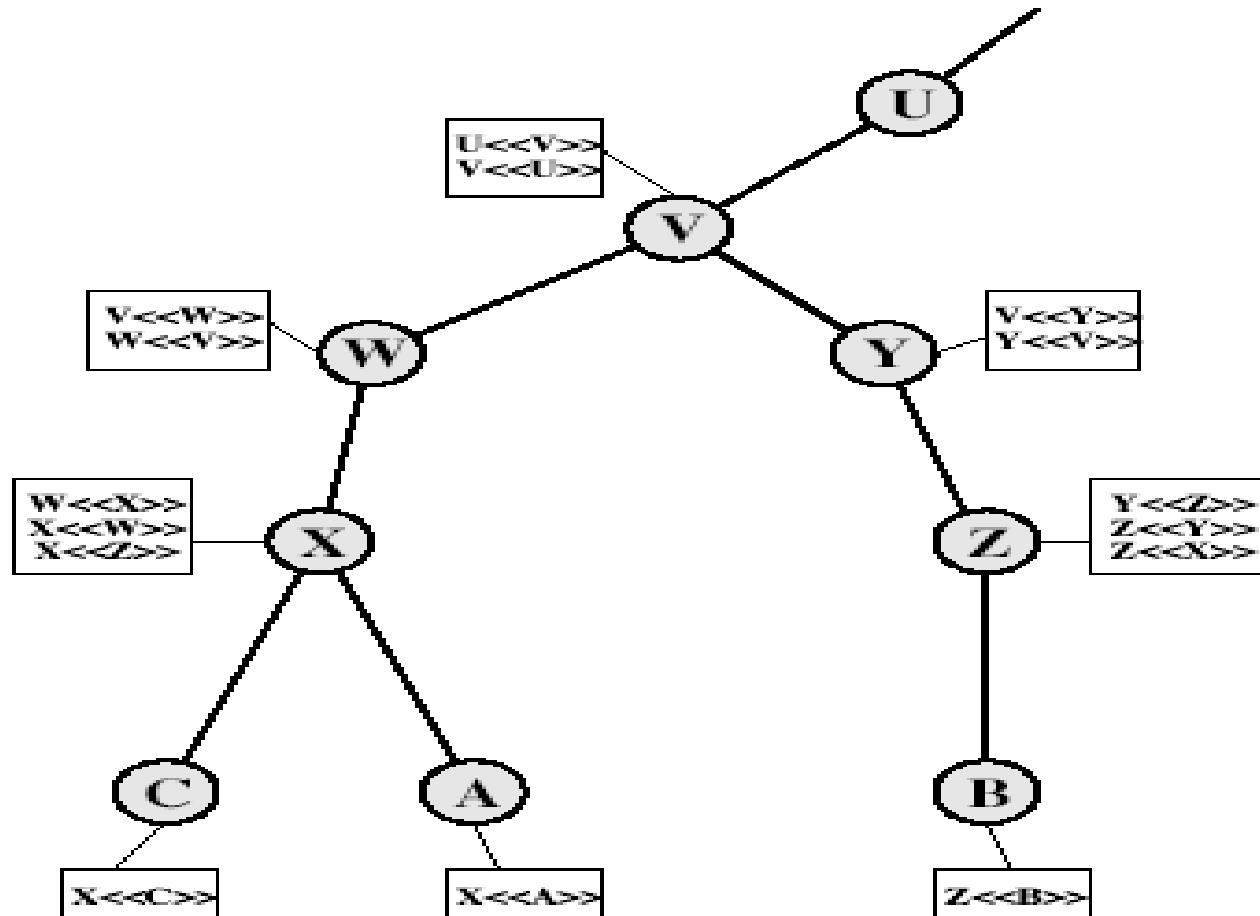
Obtaining a Certificate

- 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:
 1. user's private key is compromised
 2. user is no longer certified by this CA
 3. CA's certificate is compromised
- CA's maintain list of revoked certificates
 - the Certificate Revocation List (CRL)
- users should check certs 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

Two-Way Authentication

- 2 messages ($A \rightarrow B$, $B \rightarrow A$) 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

Three-Way Authentication

- 3 messages ($A \rightarrow B$, $B \rightarrow A$, $A \rightarrow B$) 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

Summary

- have considered:
 - Kerberos trusted key server system
 - X.509 authentication and certificates

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 15 – Electronic Mail Security

Despite the refusal of VADM Poindexter and LtCol North to appear, the Board's access to other sources of information filled much of this gap. The FBI provided documents taken from the files of the National Security Advisor and relevant NSC staff members, including messages from the PROF system between VADM Poindexter and LtCol North. The PROF messages were conversations by computer, written at the time events occurred and presumed by the writers to be protected from disclosure. In this sense, they provide a first-hand, contemporaneous account of events.

—The Tower Commission Report to President Reagan on the Iran-Contra Affair, 1987

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
- available on Unix, PC, Macintosh and Amiga systems
- originally free, now have commercial versions available also

PGP Operation – Authentication

1. sender creates a message
2. SHA-1 used to generate 160-bit hash code of message
3. hash code is encrypted with RSA using the sender's private key, and result is attached to message
4. receiver uses RSA or DSS with sender's public key to decrypt and recover hash code
5. receiver generates new hash code for message and compares with decrypted hash code, if match, message is accepted as authentic

PGP Operation – Confidentiality

1. sender generates message and random 128-bit number to be used as session key for this message only
2. message is encrypted, using CAST-128 / IDEA/3DES with session key
3. session key is encrypted using RSA with recipient's public key, then attached to message
4. receiver uses RSA with its private key to decrypt and recover session key
5. session key is used to decrypt message

PGP Operation – Confidentiality & Authentication

- uses both services on same message
 - create signature & attach to message
 - encrypt both message & signature
 - attach RSA encrypted session key

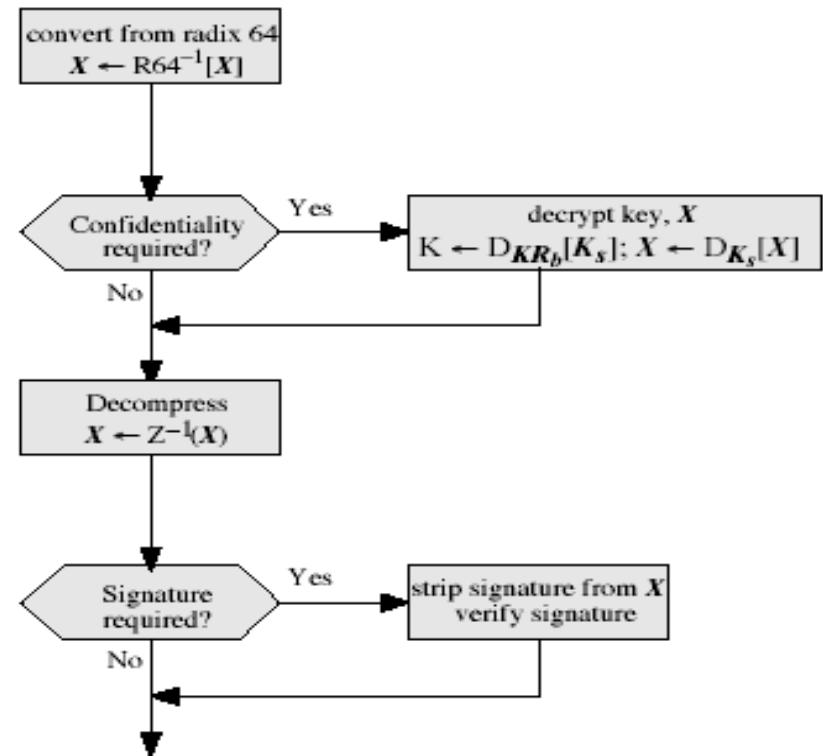
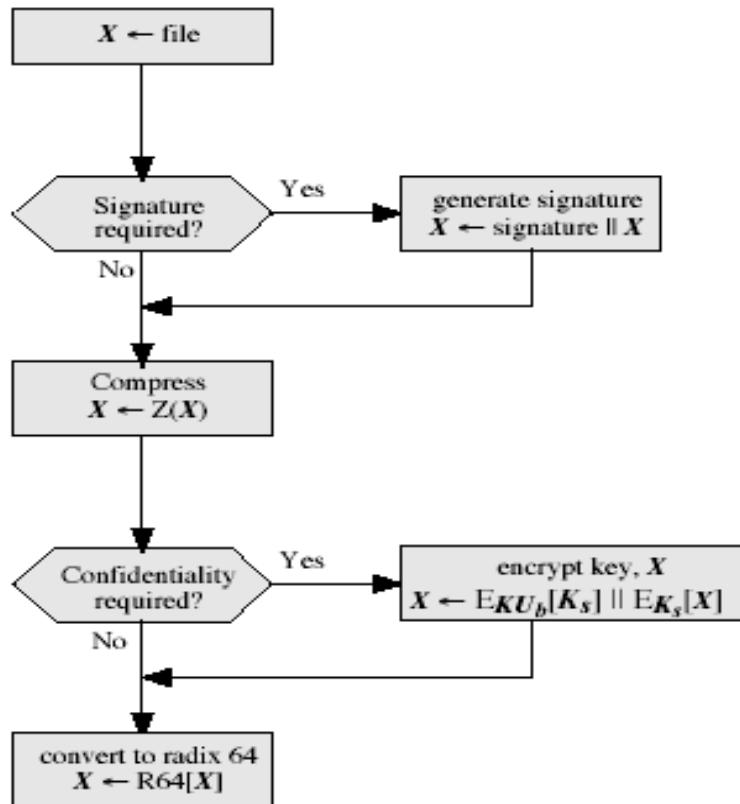
PGP Operation – Compression

- 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 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

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 Extensions)

- 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 various modern mail agents: MS Outlook, Netscape 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

- hash functions: SHA-1 & MD5
- digital signatures: DSS & RSA
- session key encryption: ElGamal & RSA
- message encryption: Triple-DES, RC2/40 and others
- have a procedure to decide which algorithms to use

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
- with 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

Summary

- have considered:
 - secure email
 - PGP
 - S/MIME

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 16 – IP Security

If a secret piece of news is divulged by a spy before the time is ripe, he must be put to death, together with the man to whom the secret was told.

—The Art of War, Sun Tzu

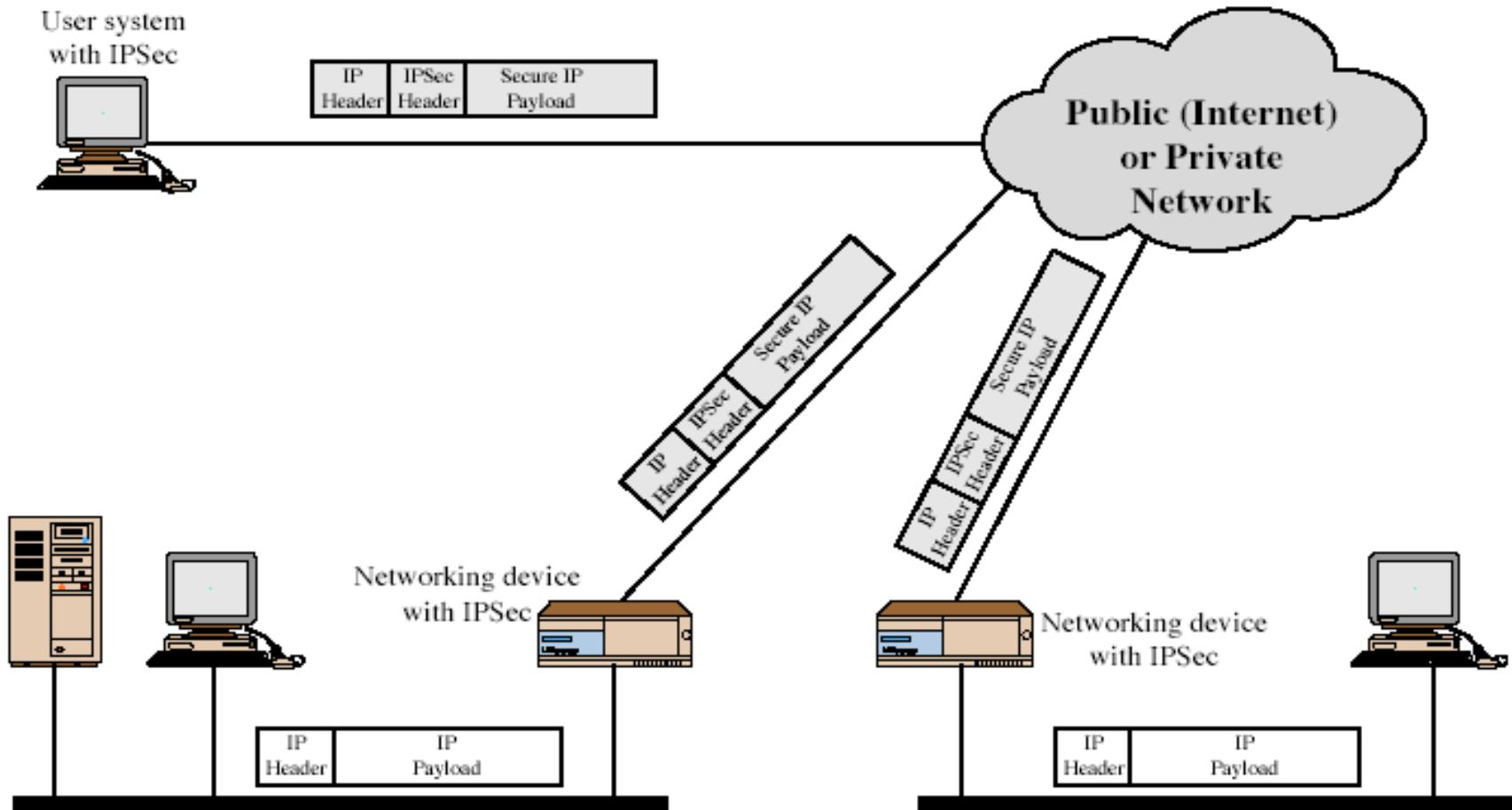
IP Security

- have considered some 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



Benefits of IPSec

- in a firewall/router provides strong security to all traffic crossing the perimeter
- is resistant to bypass
- is below transport layer, hence transparent to applications
- can be transparent to end users
- can provide security for individual users if desired

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

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

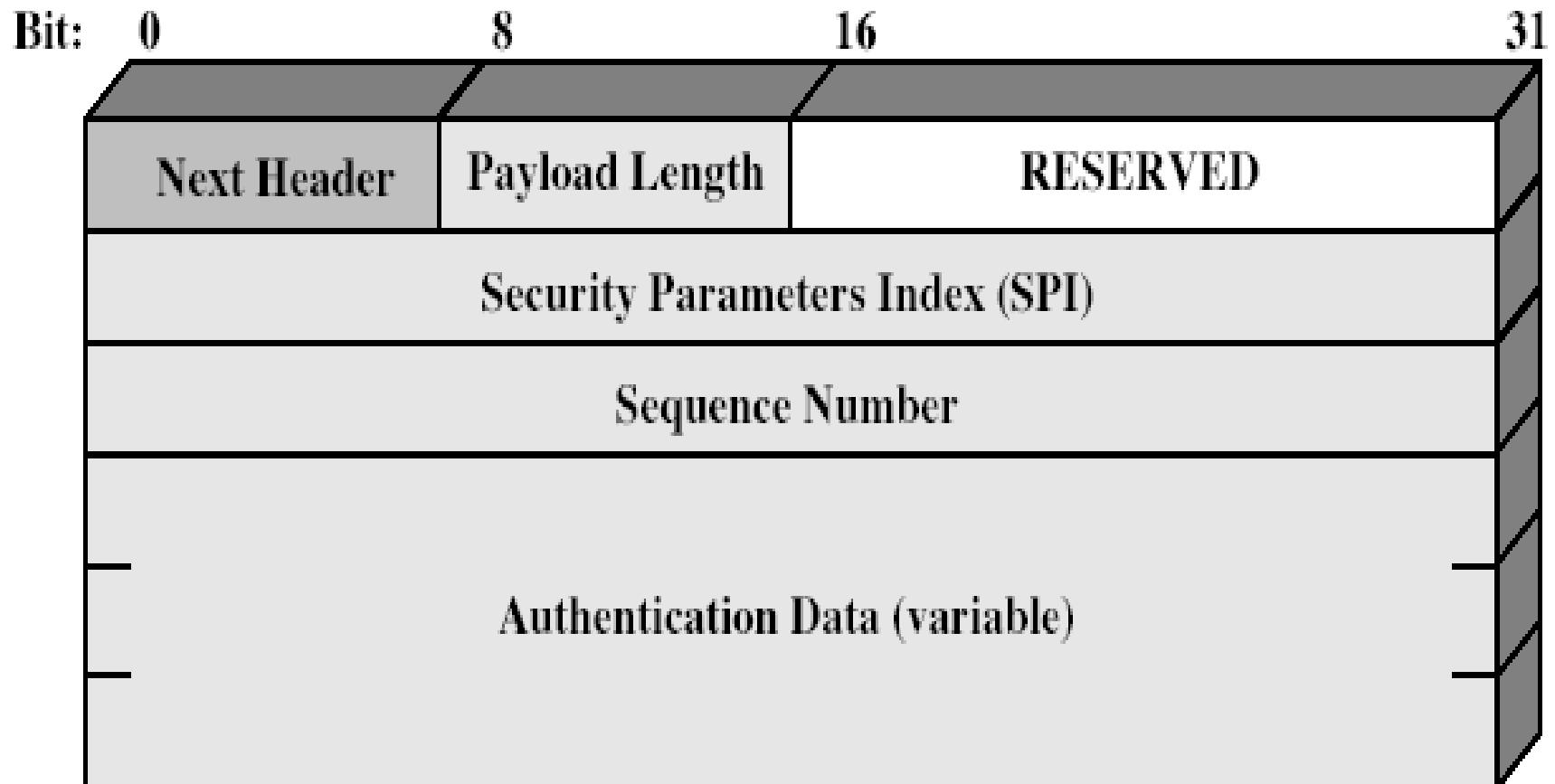
Security Associations

- 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

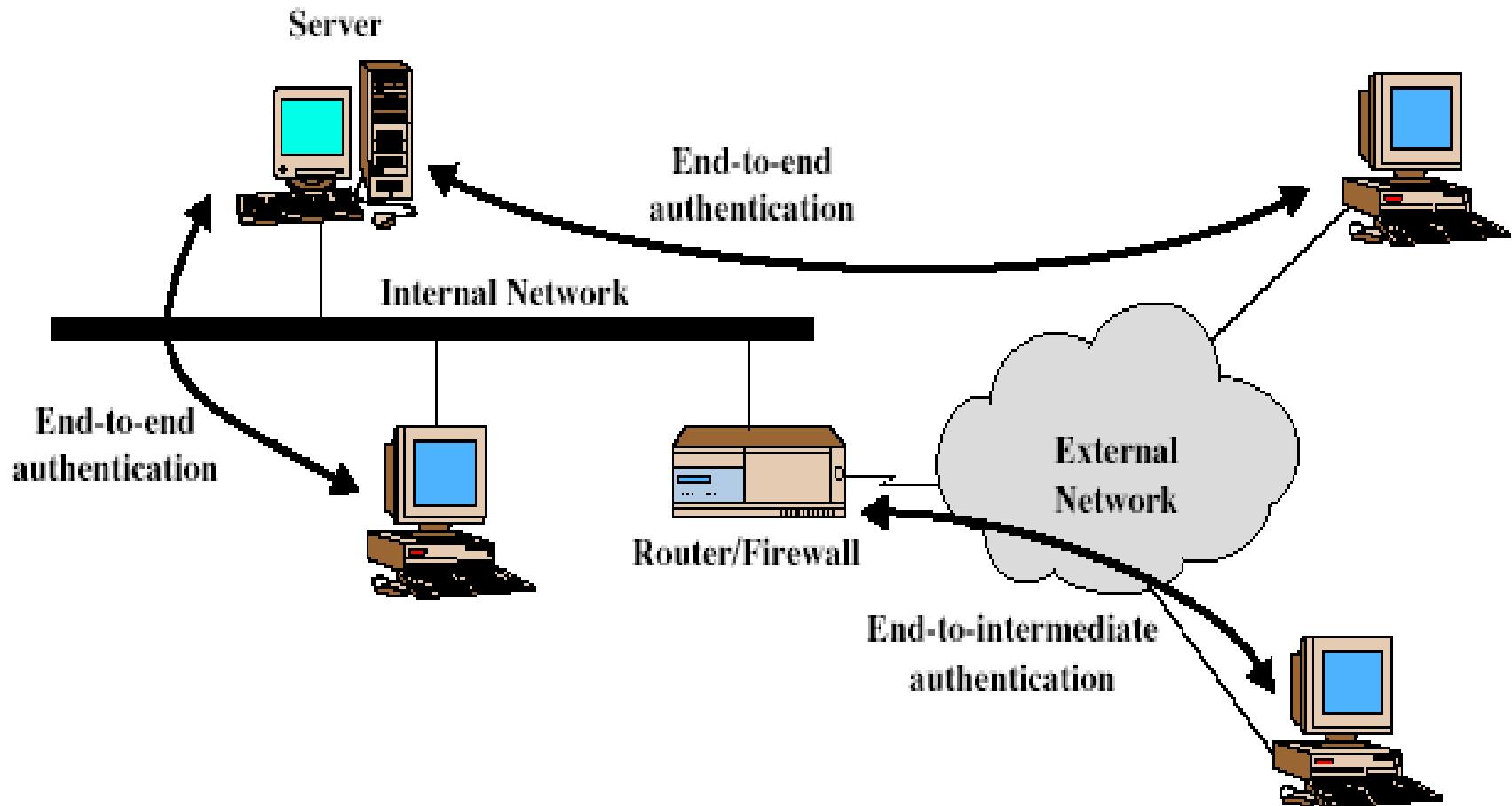
Authentication Header (AH)

- 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



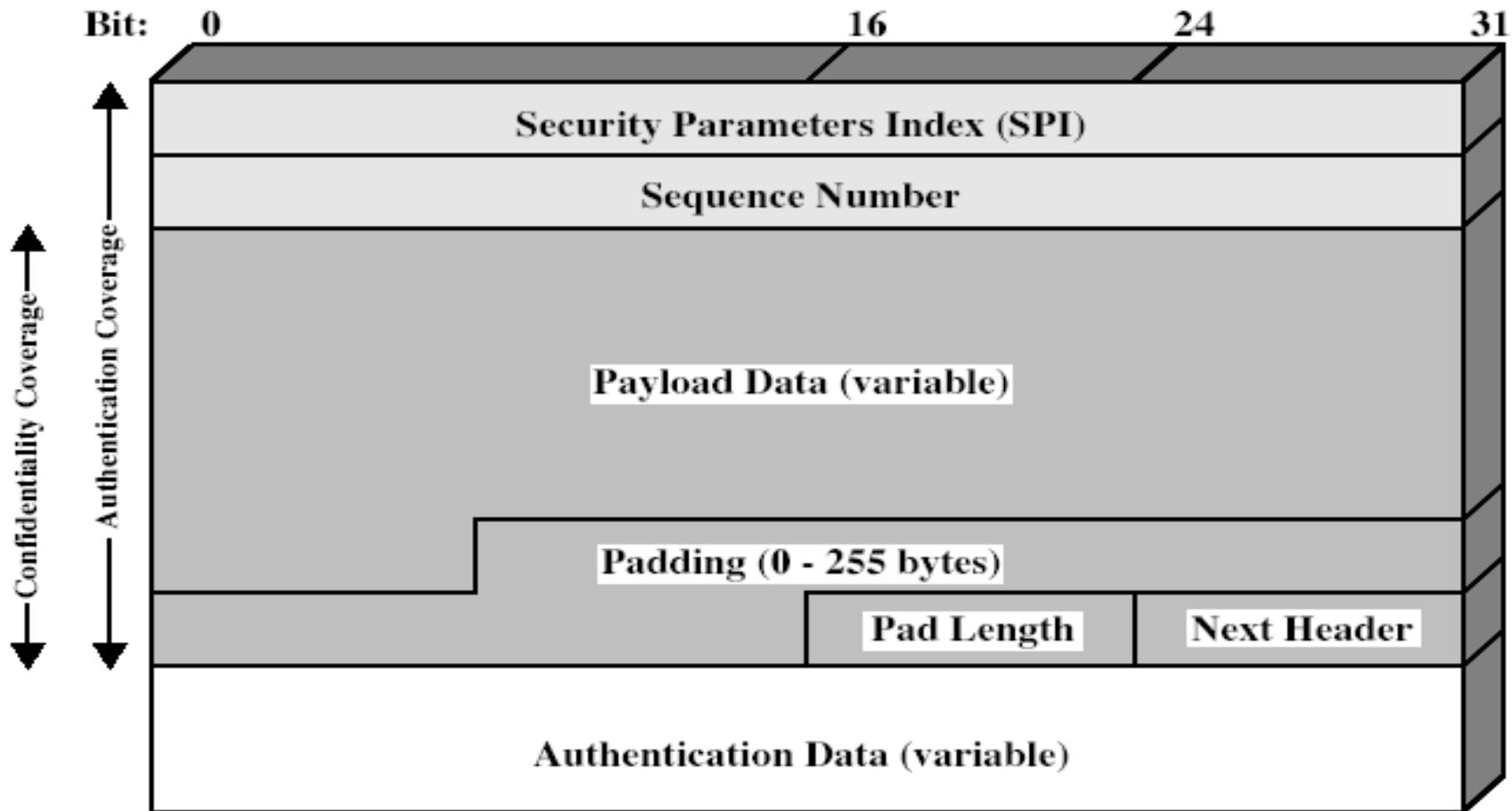
Transport & Tunnel Modes



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 most common
 - pad to meet blocksize, for traffic flow

Encapsulating Security Payload



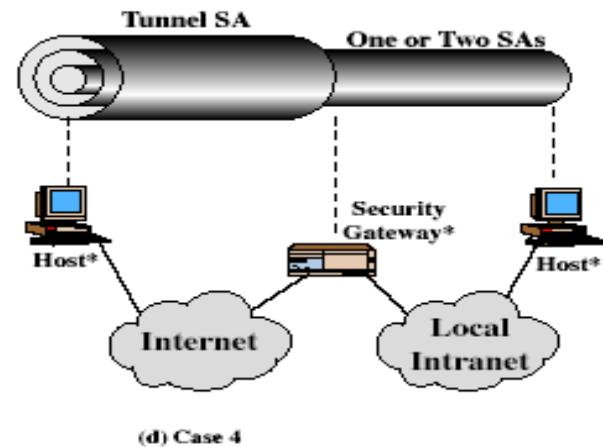
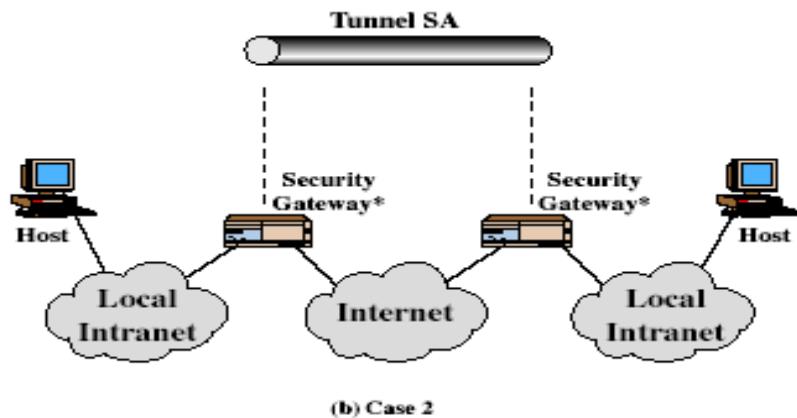
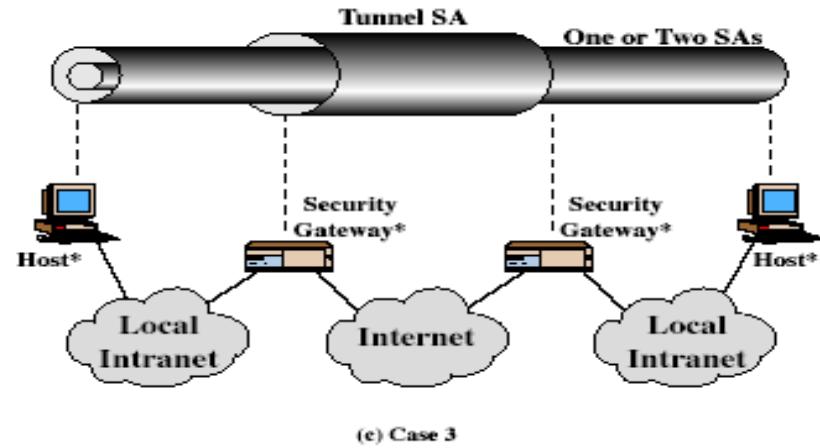
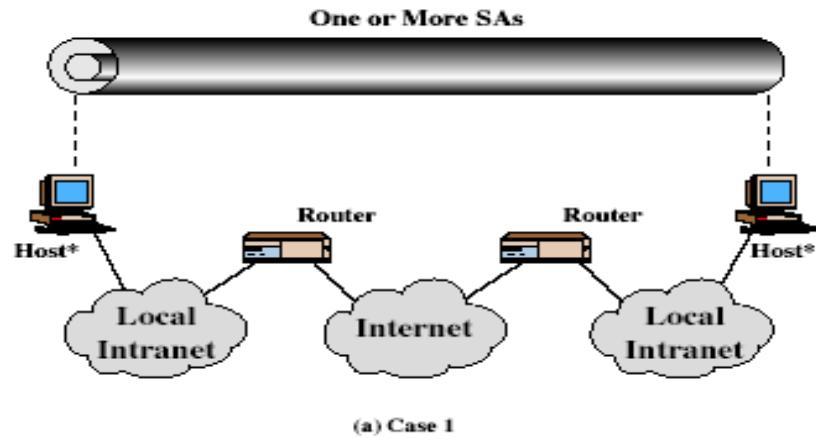
Transport vs Tunnel Mode ESP

- 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 bundle
- have 4 cases (see next)

Combining Security Associations



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

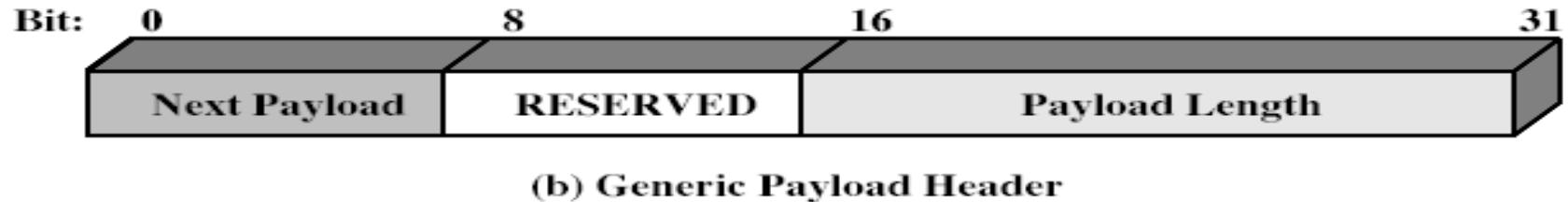
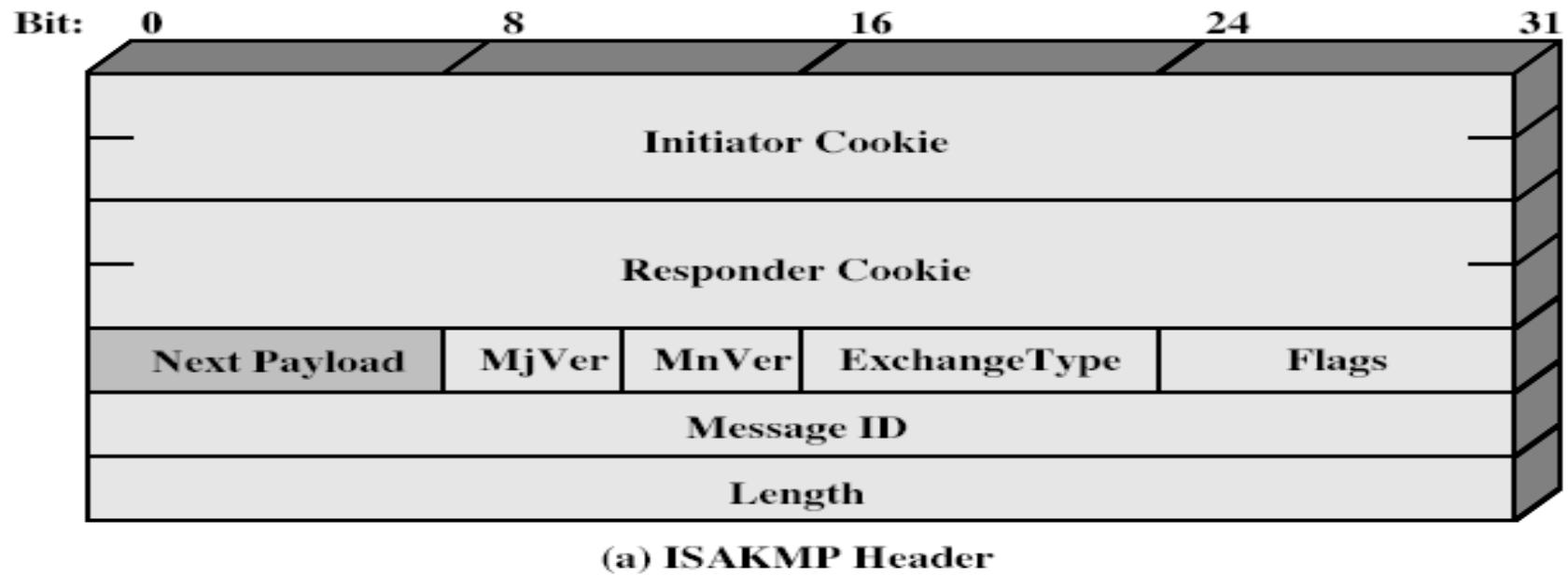
Oakley

- 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



Summary

- have considered:
 - IPSec security framework
 - AH
 - ESP
 - key management & Oakley/ISAKMP

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 17 – Web Security

Use your mentality

Wake up to reality

—From the song, "I've Got You under My Skin“ by Cole Porter

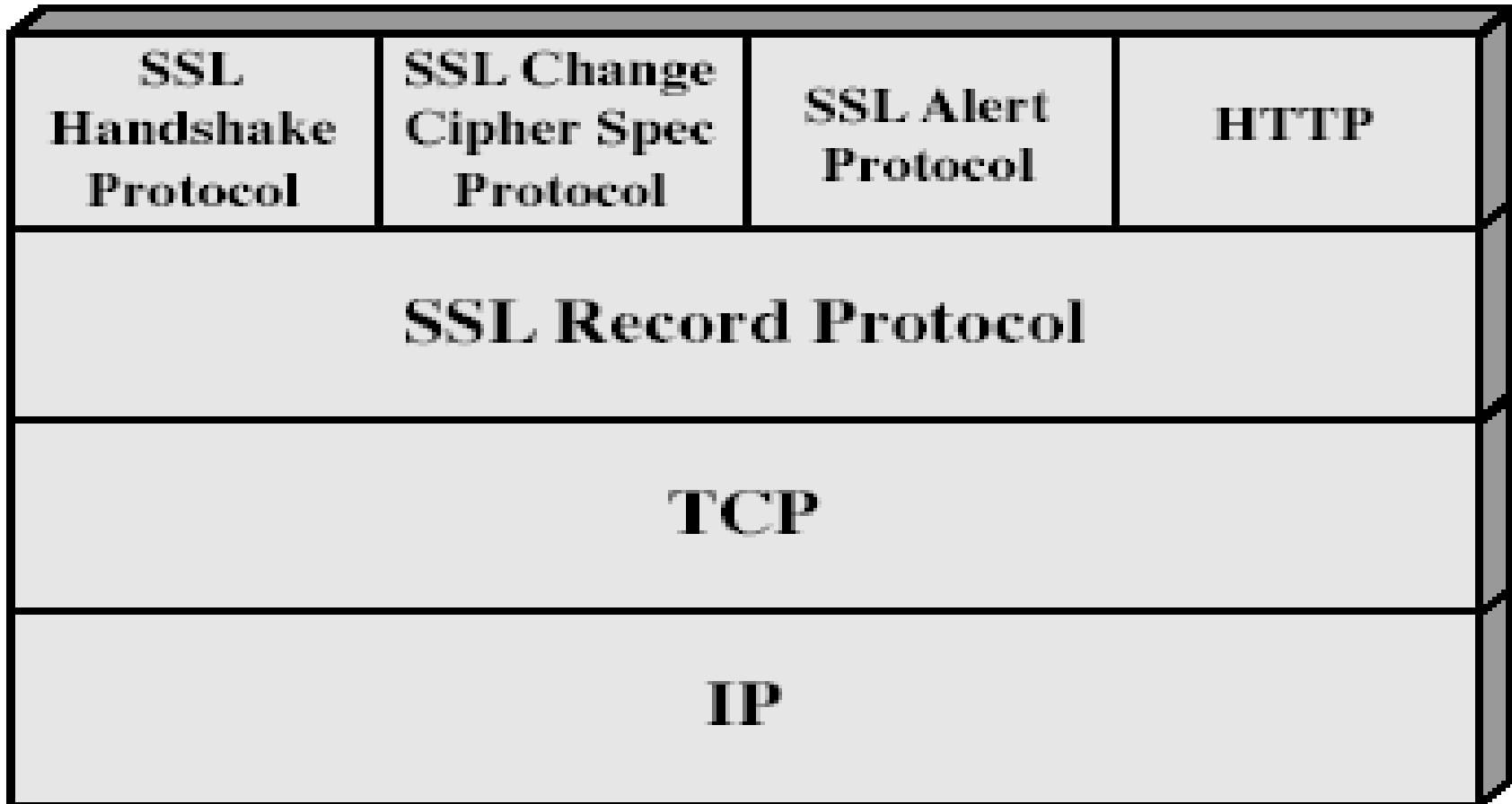
Web Security

- 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 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 connection**
 - a transient, peer-to-peer, communications link
 - associated with 1 SSL session

SSL Record Protocol

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

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

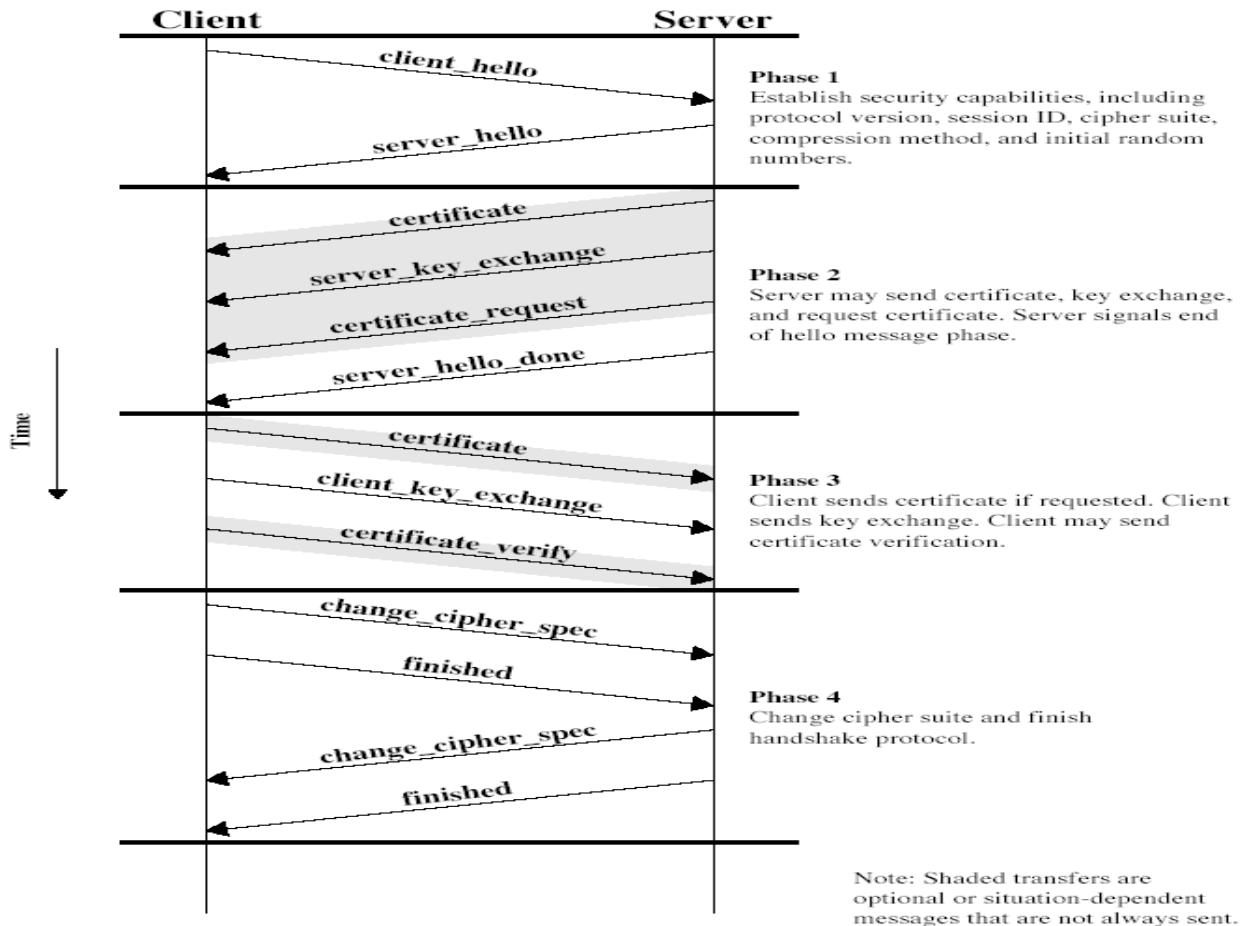
SSL Alert Protocol

- conveys SSL-related alerts to peer entity
- severity
 - warning or fatal
- specific alert
 - unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
 - 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



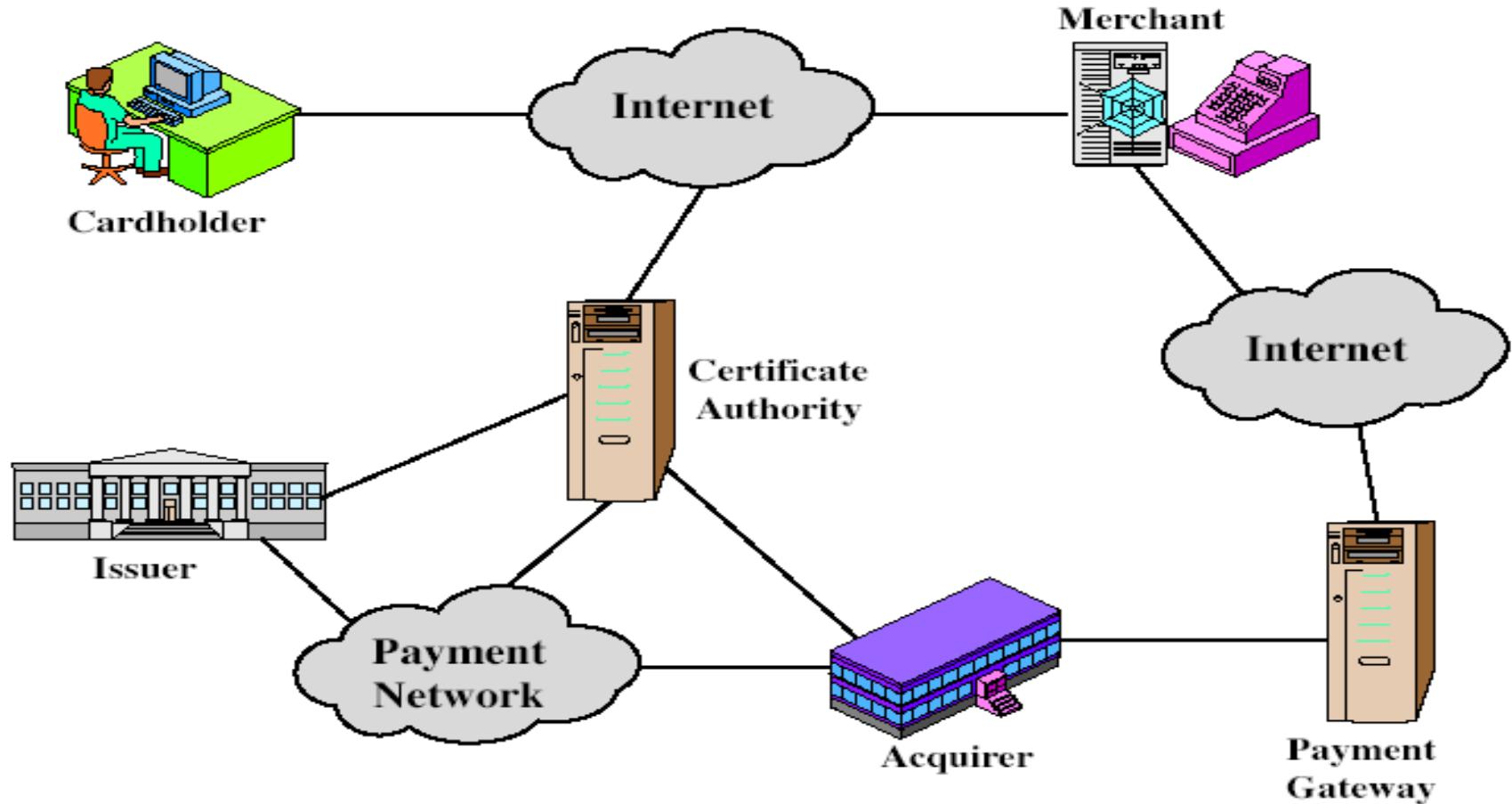
TLS (Transport Layer Security)

- 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 negotiations
 - changes in use of 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



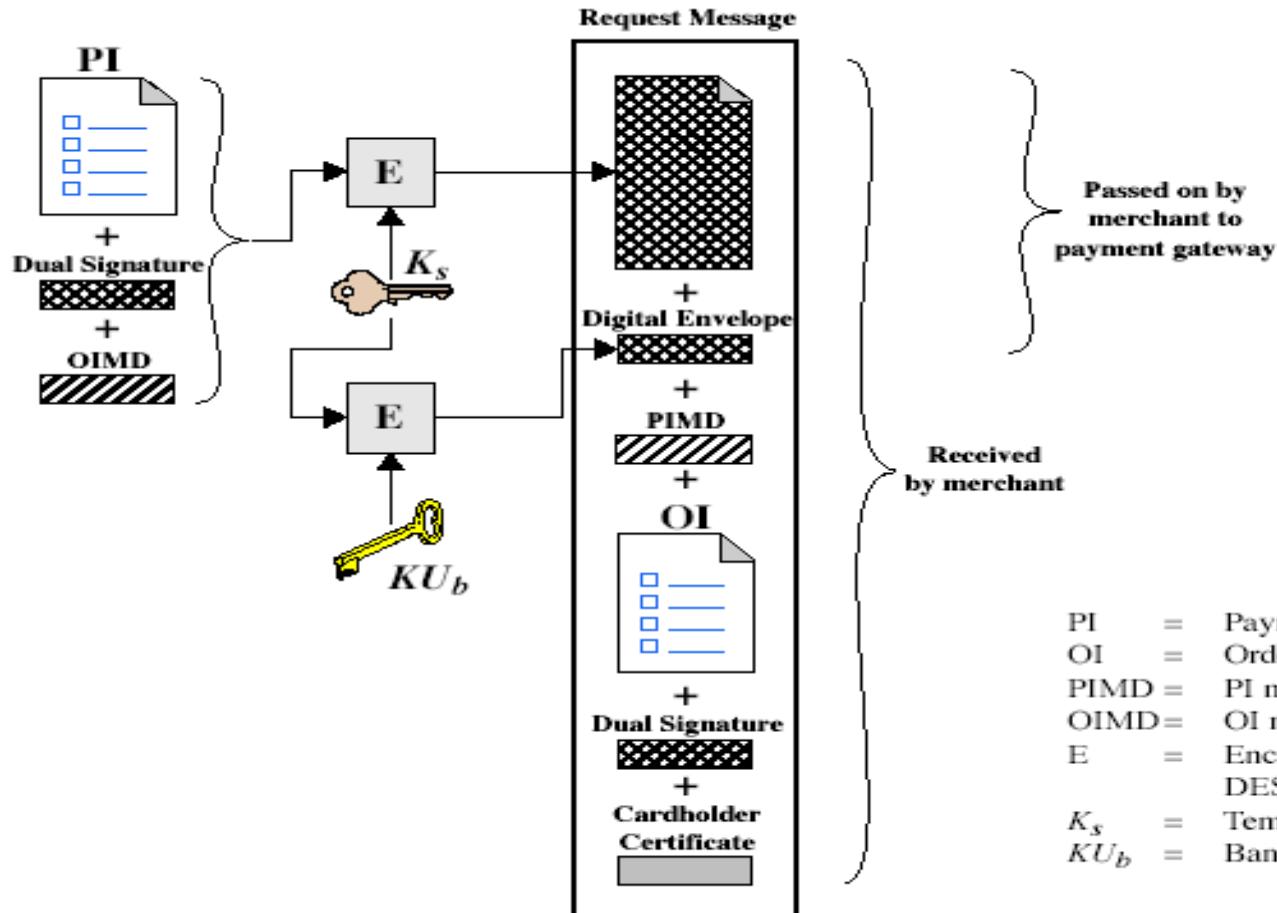
SET Transaction

1. customer opens account
2. customer receives a certificate
3. merchants have their own certificates
4. customer places an order
5. merchant is verified
6. order and payment are sent
7. merchant requests payment authorization
8. merchant confirms order
9. merchant provides goods or service
10. 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

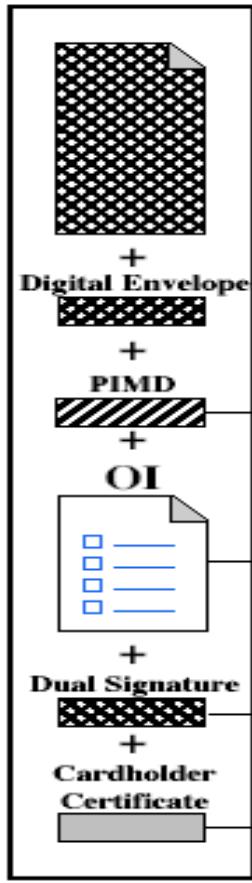
Purchase Request – Customer



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
K_{Ub}	=	Bank's public key-exchange key

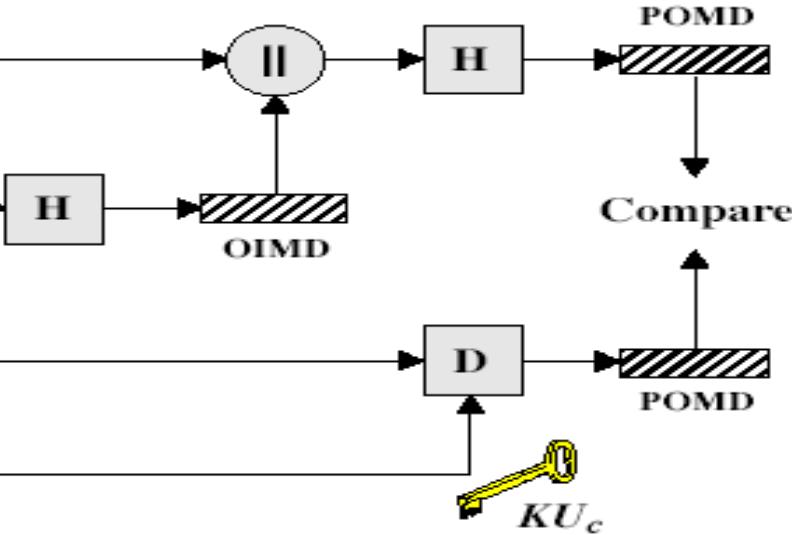
Purchase Request – Merchant

Request Message



Passed on by
merchant to
payment gateway

OI	= Order Information
OIMD	= OI message digest
POMD	= Payment Order message digest
D	= Decryption (RSA)
H	= Hash function (SHA-1)
KU_c	= Customer's public signature key



Purchase Request – Merchant

1. verifies cardholder certificates using CA sigs
2. 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
3. processes order and forwards the payment information to the payment gateway for authorization (described later)
4. sends a purchase response to cardholder

Payment Gateway Authorization

1. verifies all certificates
2. decrypts digital envelope of authorization block to obtain symmetric key & then decrypts authorization block
3. verifies merchant's signature on authorization block
4. decrypts digital envelope of payment block to obtain symmetric key & then decrypts payment block
5. verifies dual signature on payment block
6. verifies that transaction ID received from merchant matches that in PI received (indirectly) from customer
7. requests & receives an authorization from issuer
8. sends authorization response back to merchant

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

Summary

- have considered:
 - need for web security
 - SSL/TLS transport layer security protocols
 - SET secure credit card payment protocols

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 18 – Intruders

They agreed that Graham should set the test for Charles Mabledene. It was neither more nor less than that Dragon should get Stern's code. If he had the 'in' at Utting which he claimed to have this should be possible, only loyalty to Moscow Centre would prevent it. If he got the key to the code he would prove his loyalty to London Central beyond a doubt.

—***Talking to Strange Men, Ruth Rendell***

Intruders

- significant issue for networked systems is hostile or unwanted access
- either via network or local
- can identify classes of intruders:
 - masquerader
 - misfeasor
 - clandestine user
- varying levels of competence

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

Intrusion Techniques

- aim to increase privileges on 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
 - try default passwords shipped with systems
 - try all short passwords
 - then try by searching dictionaries of common words
 - intelligent searches try passwords associated with the user (variations on names, birthday, phone, common words/interests)
 - before exhaustively searching all possible passwords
- check by login attempt 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
 - 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
 - compare audit records or states against rules

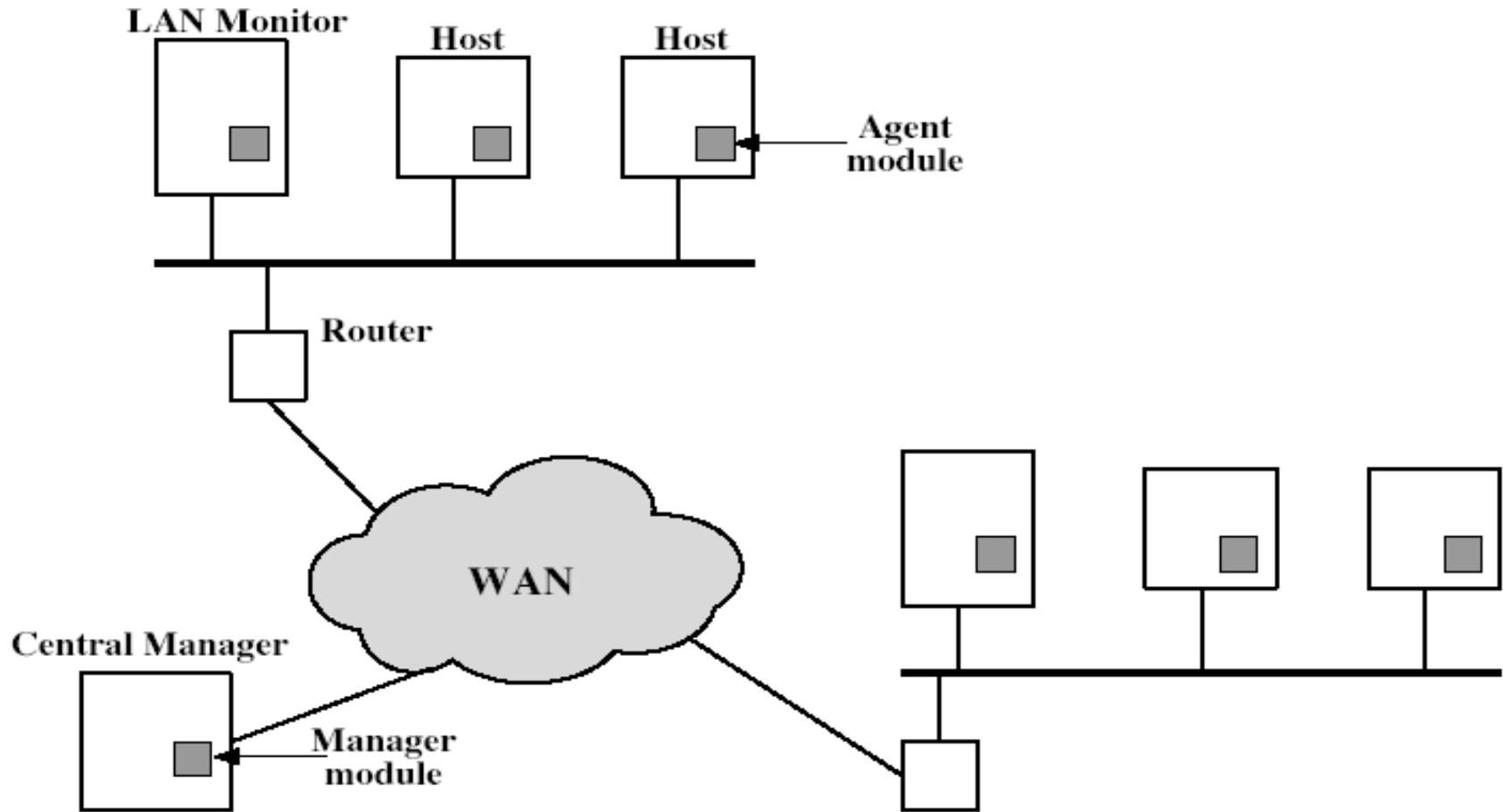
Base-Rate Fallacy

- 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

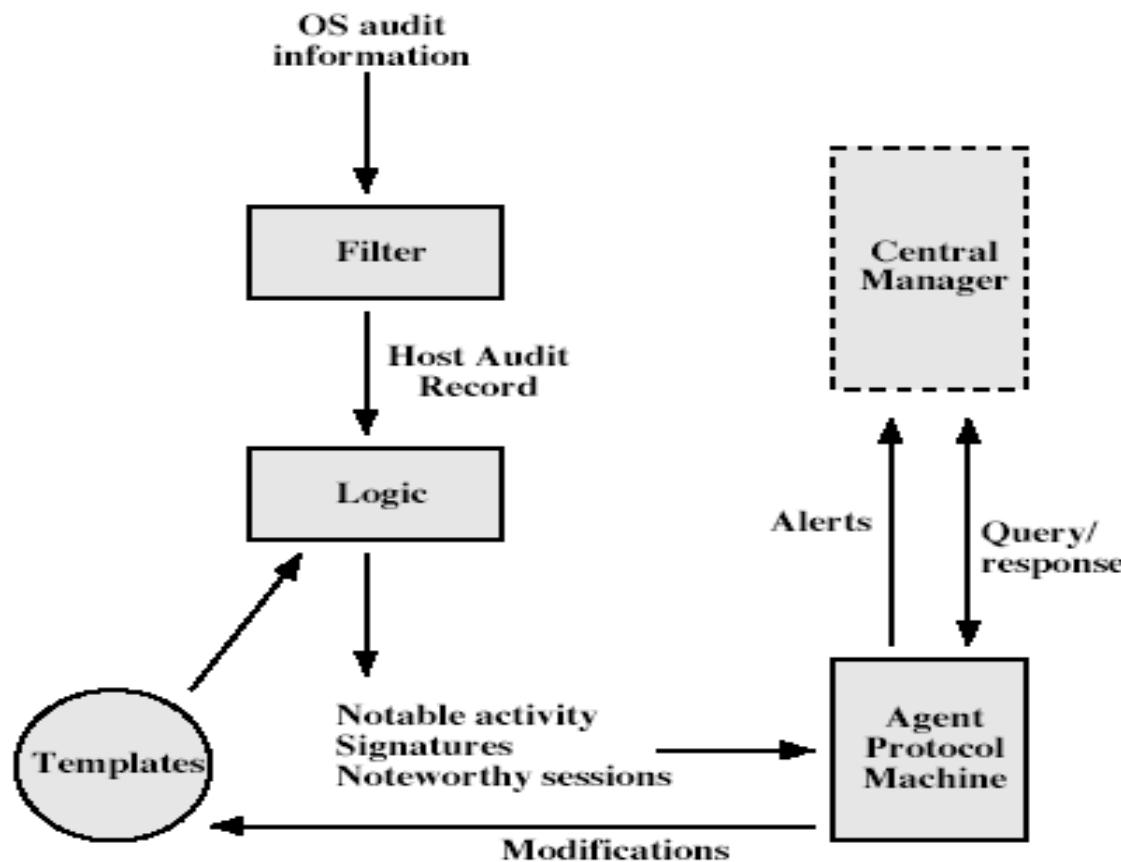
Distributed Intrusion Detection

- 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
- may be single or multiple networked systems

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

Managing Passwords

- need policies and good user education
- ensure **every** account has a default password
- ensure users change the default passwords to something they can remember
- protect password file from general access
- set technical policies to enforce good passwords
 - minimum length (>6)
 - require a mix of upper & lower case letters, numbers, punctuation
 - block known dictionary words

Managing Passwords

- may reactively run password guessing tools
 - note that good dictionaries exist for almost any language/interest group
- may enforce periodic changing of passwords
- have system monitor failed login attempts, & lockout account if see too many in a short period
- do need to educate users and get support
- balance requirements with user acceptance
- be aware of **social engineering** attacks

Proactive Password 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 previous slide)
 - compare against dictionary of bad passwords
 - use algorithmic (markov model or bloom filter) to detect poor choices

Summary

- have considered:
 - problem of intrusion
 - intrusion detection (statistical & rule-based)
 - password management

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 19 – Malicious Software

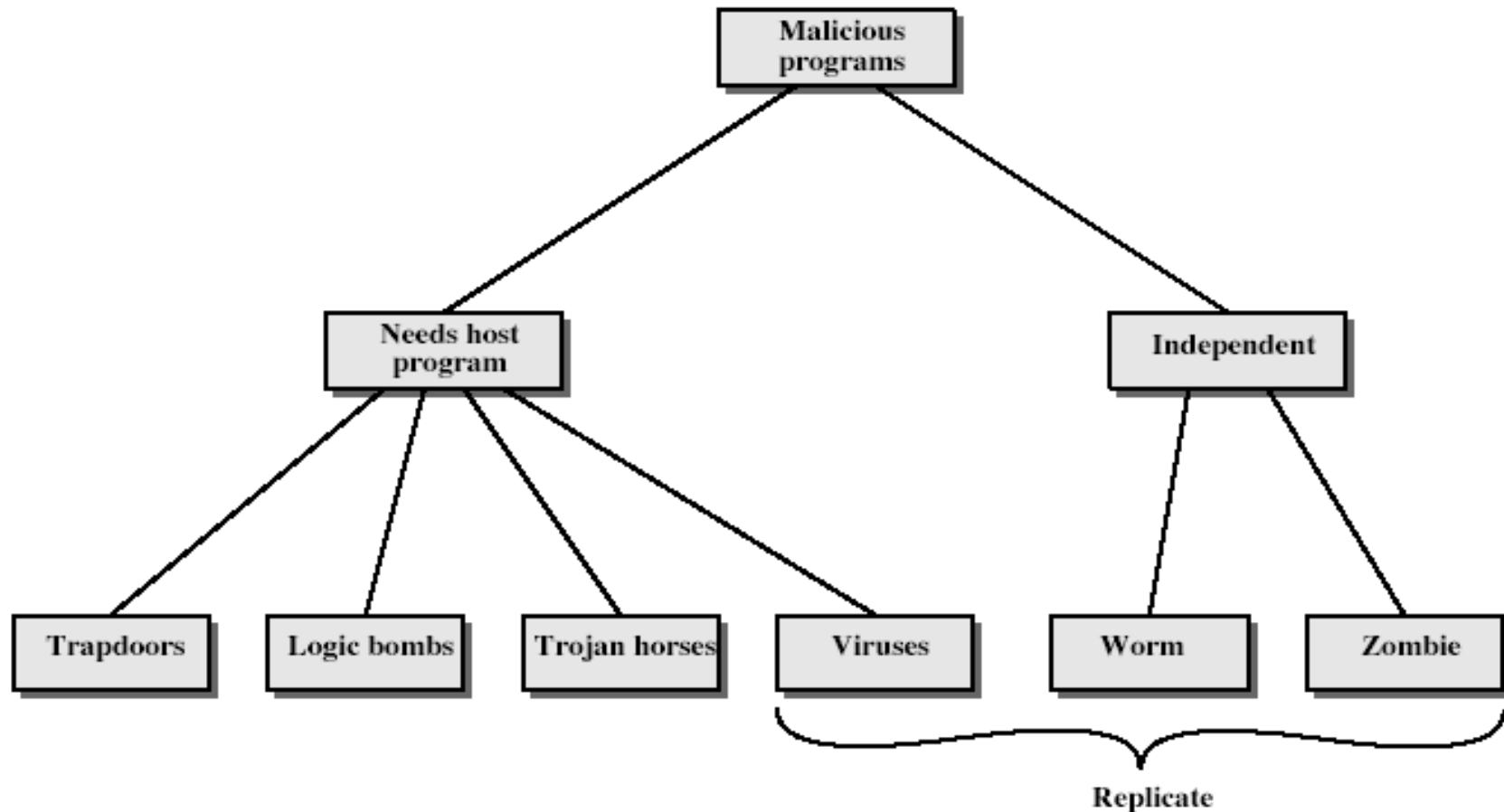
What is the concept of defense: The parrying of a blow. What is its characteristic feature: Awaiting the blow.

—On War, Carl Von Clausewitz

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



Trapdoors

- 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

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 some 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
- macro 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
- blurs distinction between data and program files making task of detection much harder
- classic trade-off: "ease of use" vs "security"

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
- usually targeted at Microsoft Outlook mail agent & Word/Excel documents

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**
 - exploited bug in MS IIS to penetrate & spread
 - 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**
 - had backdoor installed to allow remote control
- **Nimda**
 - used multiple infection mechanisms
 - email, shares, web client, IIS, Code Red 2 backdoor

Virus Countermeasures

- viral attacks exploit lack of integrity control on systems
- to defend need to add such controls
- typically by one or more of:
 - **prevention** - block virus infection mechanism
 - **detection** - of viruses in infected system
 - **reaction** - 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 program checksums 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

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

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

Summary

- have considered:
 - various malicious programs
 - trapdoor, logic bomb, trojan horse, zombie
 - viruses
 - worms
 - countermeasures

Cryptography and Network Security

Third Edition
by William Stallings

Lecture slides by Lawrie Brown

Chapter 20 – Firewalls

The function of a strong position is to make the forces holding it practically unassailable

—On War, Carl Von Clausewitz

Introduction

- seen evolution of information systems
- now everyone want to be on the Internet
- and to interconnect networks
- has persistent security concerns
 - can't easily secure every system in org
- need "harm minimisation"
- a **Firewall** usually part of this

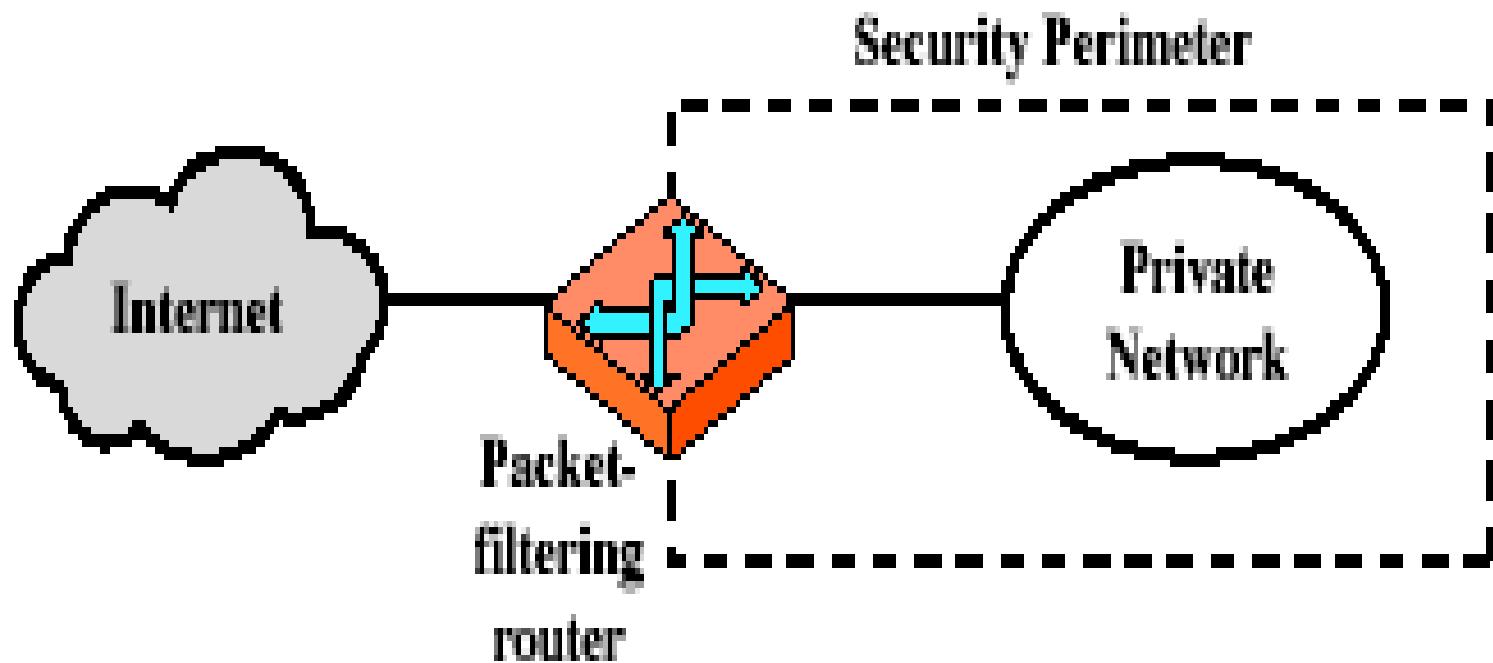
What is a Firewall?

- 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
- is itself immune to penetration
- provides **perimeter defence**

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 employee
- cannot protect against transfer of all virus infected programs or files
 - because of huge range of O/S & file types

Firewalls – Packet Filters



(a) Packet-filtering router

Firewalls – Packet Filters

- simplest of components
- 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

Table 20.1 Packet-Filtering Examples

	action	ourhost	port	theirhost	port	comment	
A	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

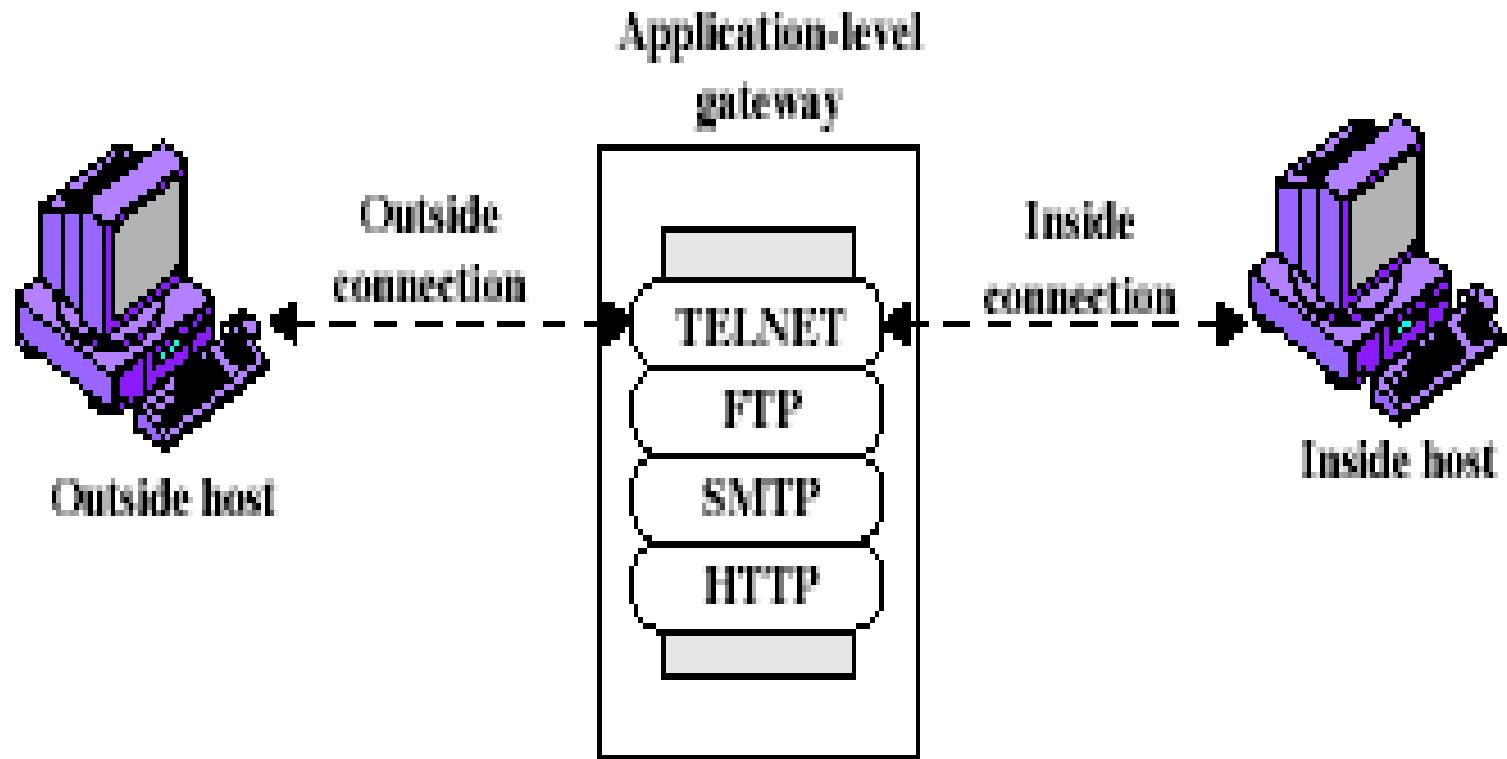
Attacks on Packet Filters

- 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

- examine each IP packet in context
 - keeps tracks of client-server sessions
 - checks each packet validly belongs to one
- better able to detect bogus packets out of context

Firewalls - Application Level Gateway (or Proxy)

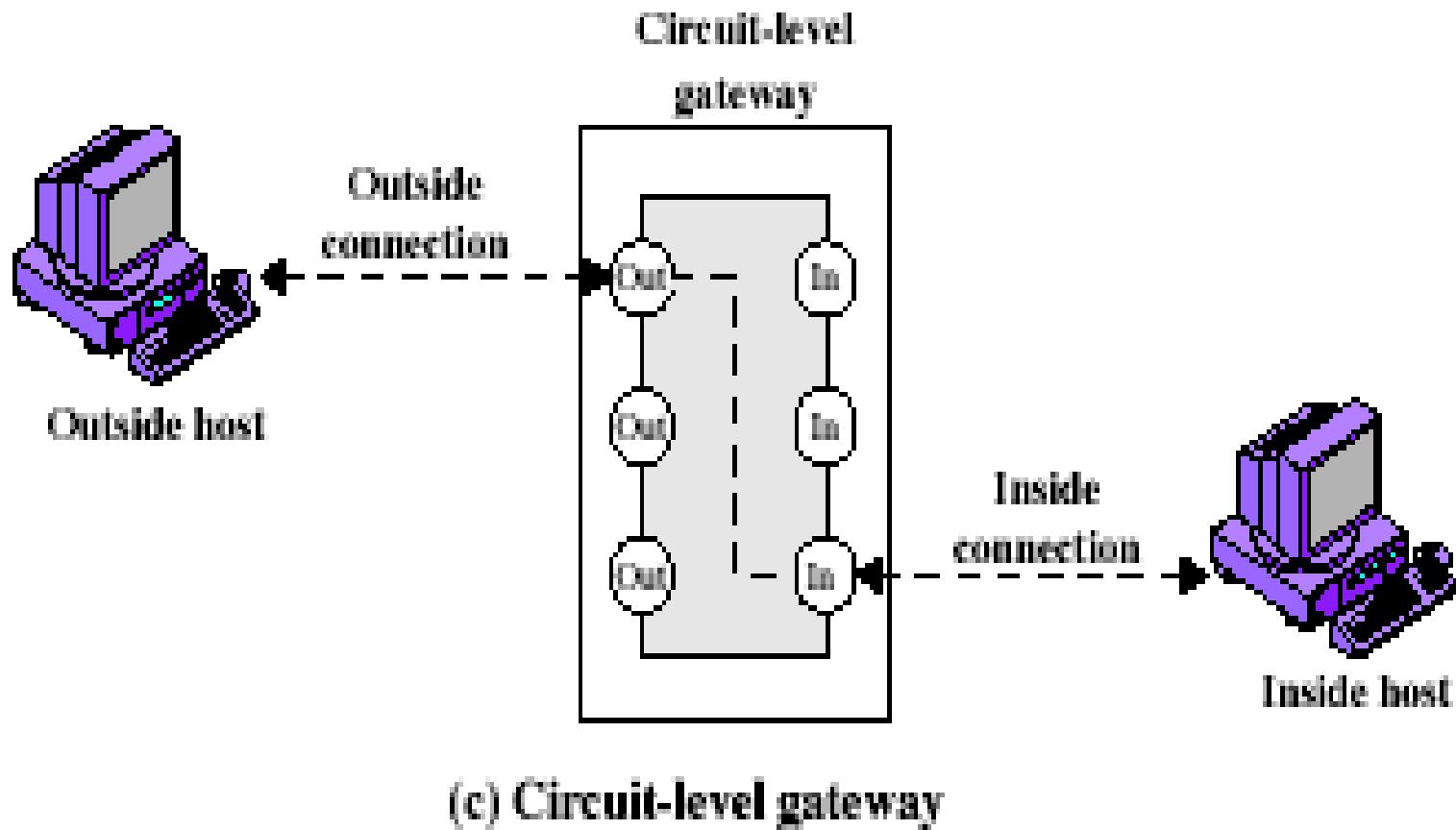


(b) Application-level gateway

Firewalls - Application Level Gateway (or Proxy)

- use an 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
- need separate proxies for each service
 - some services naturally support proxying
 - others are more problematic
 - custom services generally not supported

Firewalls - Circuit Level Gateway



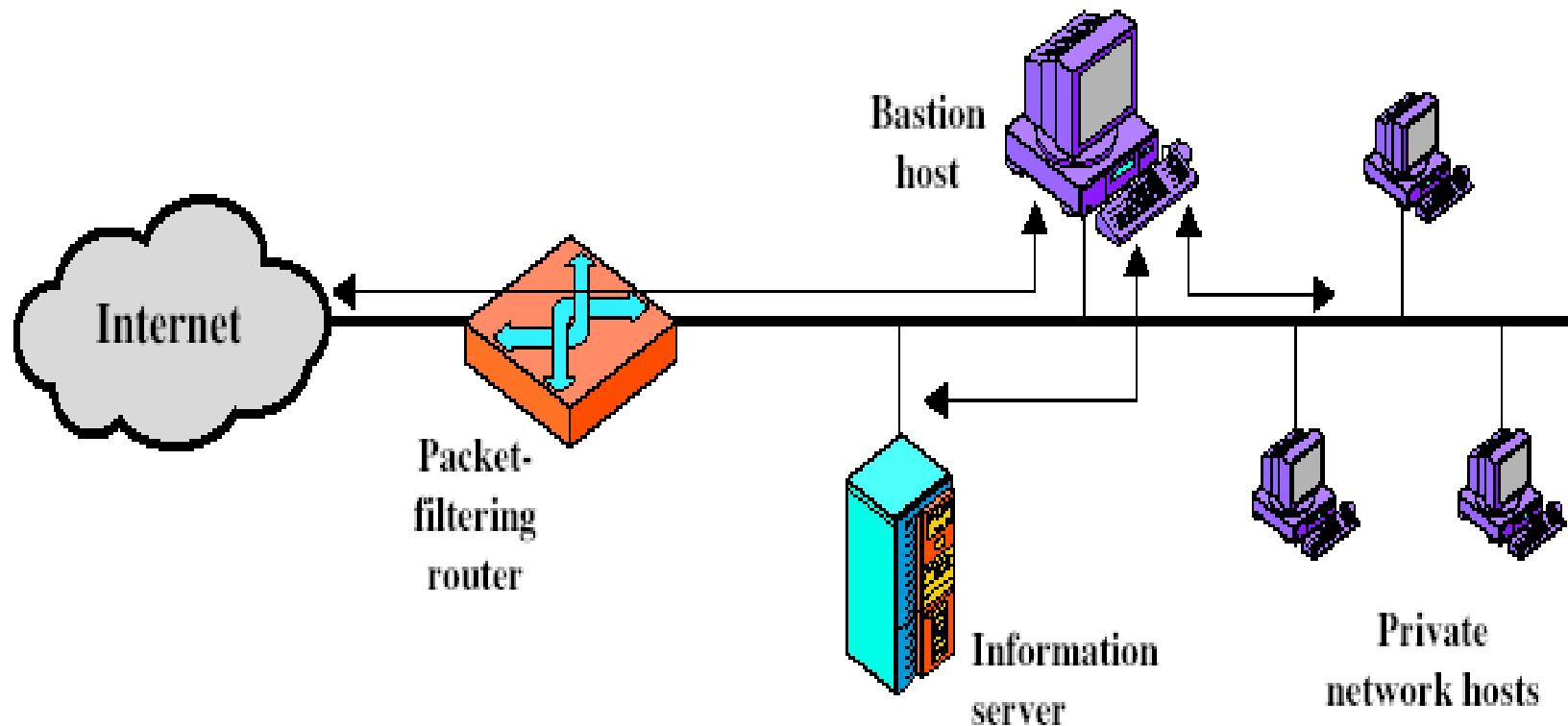
Firewalls - Circuit Level Gateway

- 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 commonly used for this

Bastion Host

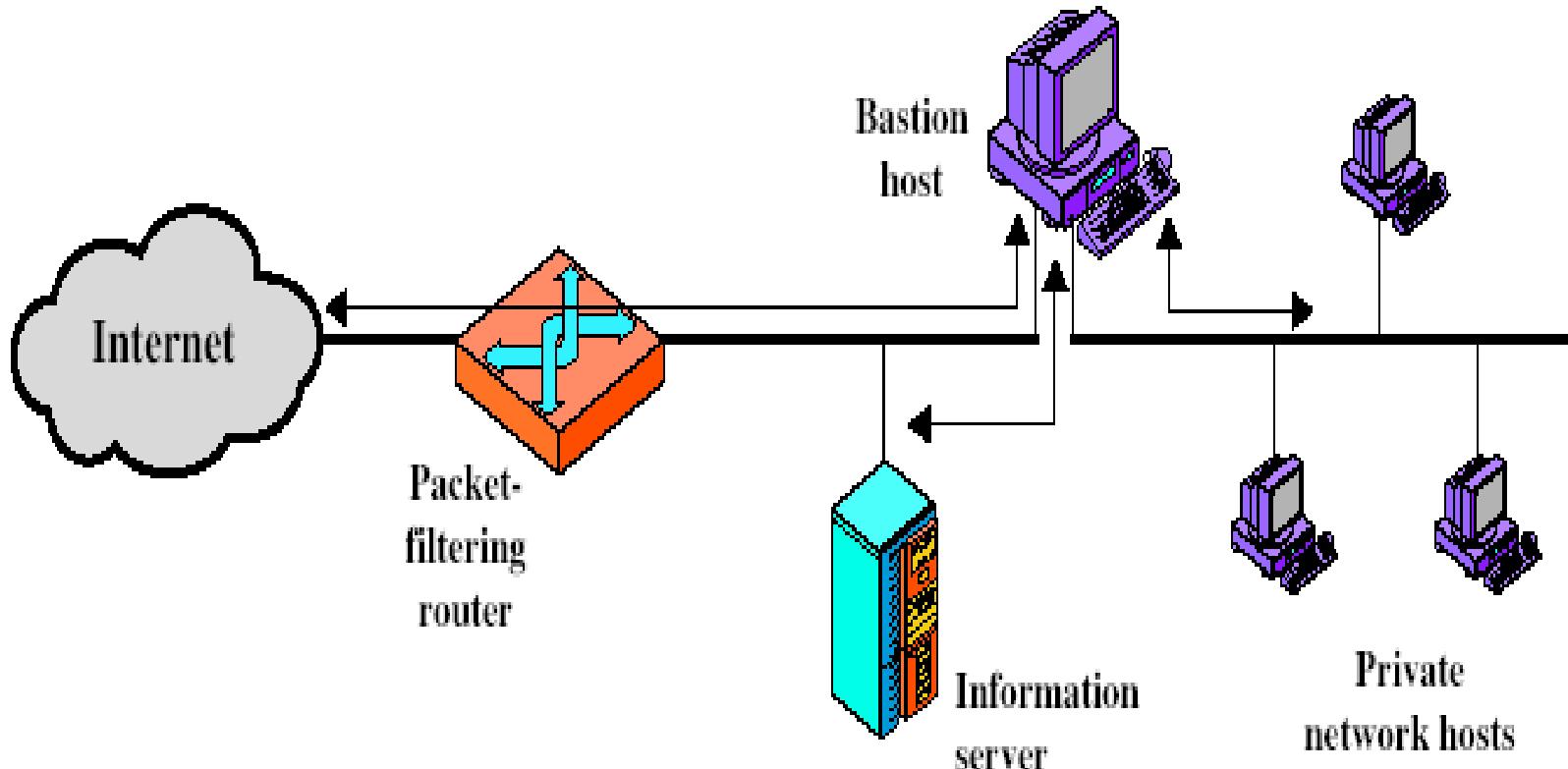
- highly secure host system
- potentially exposed to "hostile" elements
- hence is secured to withstand this
- may support 2 or more net connections
- may be trusted to enforce trusted separation between network connections
- runs circuit / application level gateways
- or provides externally accessible services

Firewall Configurations



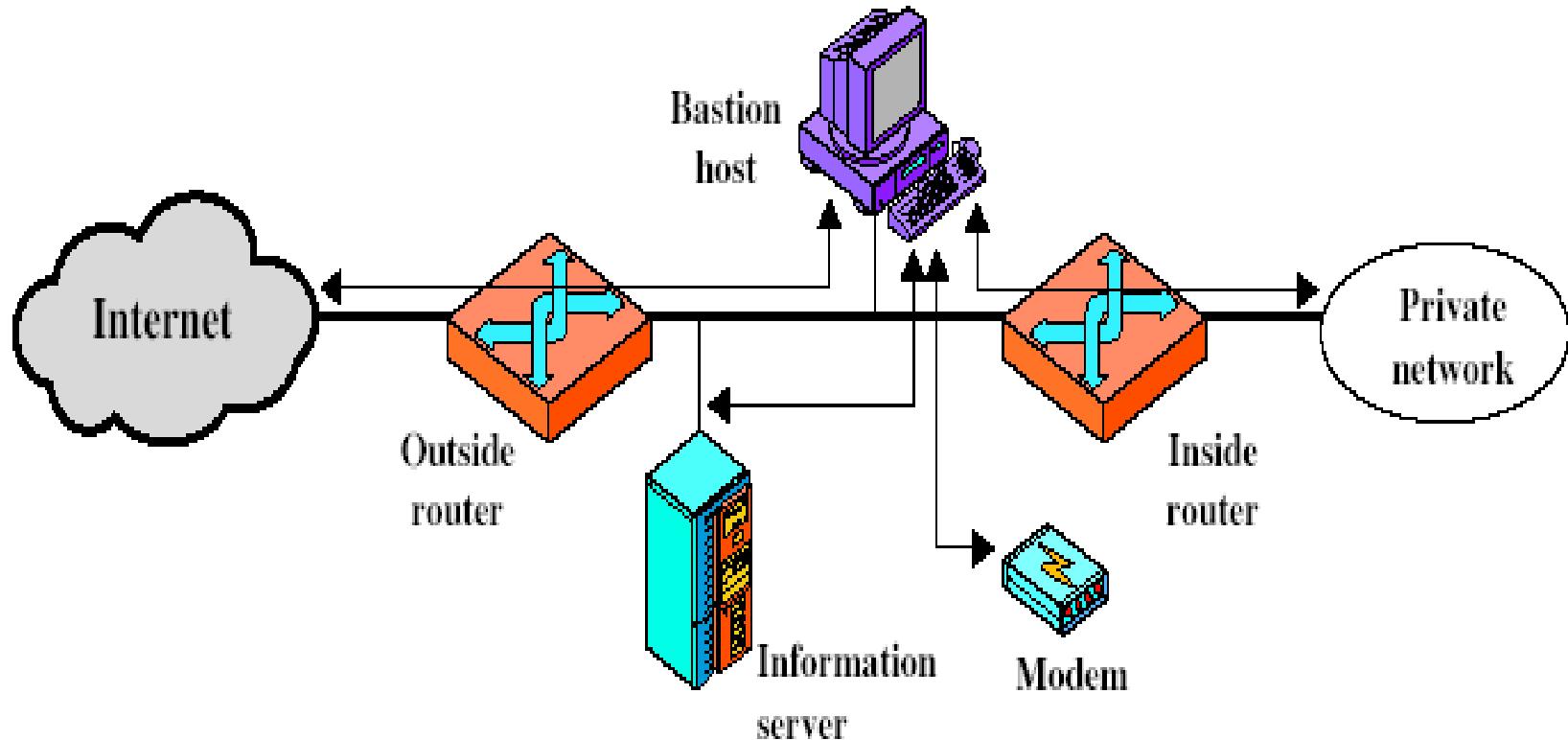
(a) Screened host firewall system (single-homed bastion host)

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		Read	
	Execute		Write	
Process2				Read
•				
•				
•				

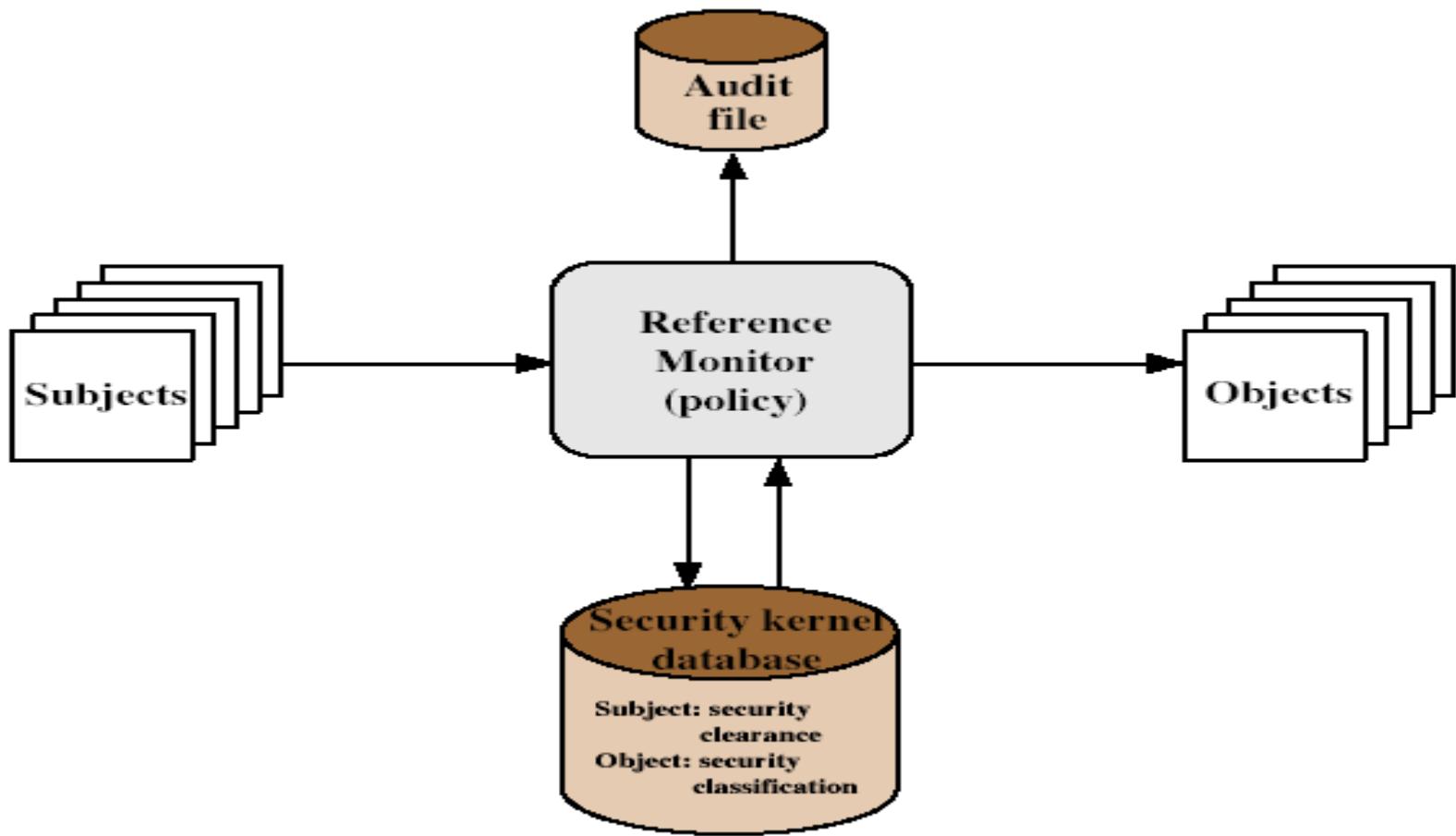
Trusted Computer Systems

- 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)
- want to consider ways of increasing confidence in systems to enforce these rights
- known as multilevel security
 - subjects have **maximum & current** security level
 - objects have a fixed security level **classification**

Bell LaPadula (BLP) Model

- one of the most famous security models
- implemented as mandatory policies on system
- has two key policies:
- **no read up** (simple security property)
 - a subject can only read/write an object if the current security level of the subject dominates (\geq) the classification of the object
- **no write down** (*-property)
 - a subject can only append/write to an object if the current security level of the subject is dominated by (\leq) the classification of the object

Reference Monitor



Evaluated Computer Systems

- governments can evaluate IT systems
- against a range of standards:
 - TCSEC, IPSEC and now Common Criteria
- define a number of “levels” of evaluation with increasingly stringent checking
- have published lists of evaluated products
 - though aimed at government/defense use
 - can be useful in industry also

Summary

- have considered:
 - firewalls
 - types of firewalls
 - configurations
 - access control
 - trusted systems