

Introduction to Basic Security Concepts

**CSC 4575/5575
Information Assurance and Cryptography
Spring 2019**

Slide Source:
Introduction to Computer Security, Matt Bishop, Addison Wesley, Chapter 1

Security Concepts

- Threat classes
 - CIA Triad
 - Defense in depth
-

Information Assurance (IA) & Security

- IA is the perception that systems are operating as expectation in an environment with protections in place.
- Security is measures and controls to achieve IA.

EXPECTATION



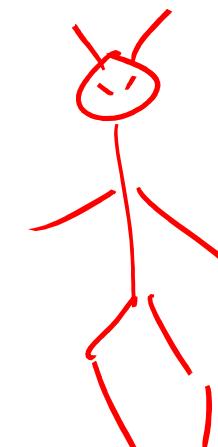
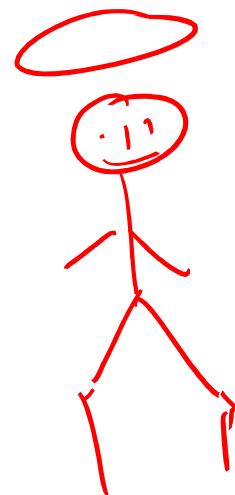
PROTECTION



PERCEPTION

Two Sides in IA

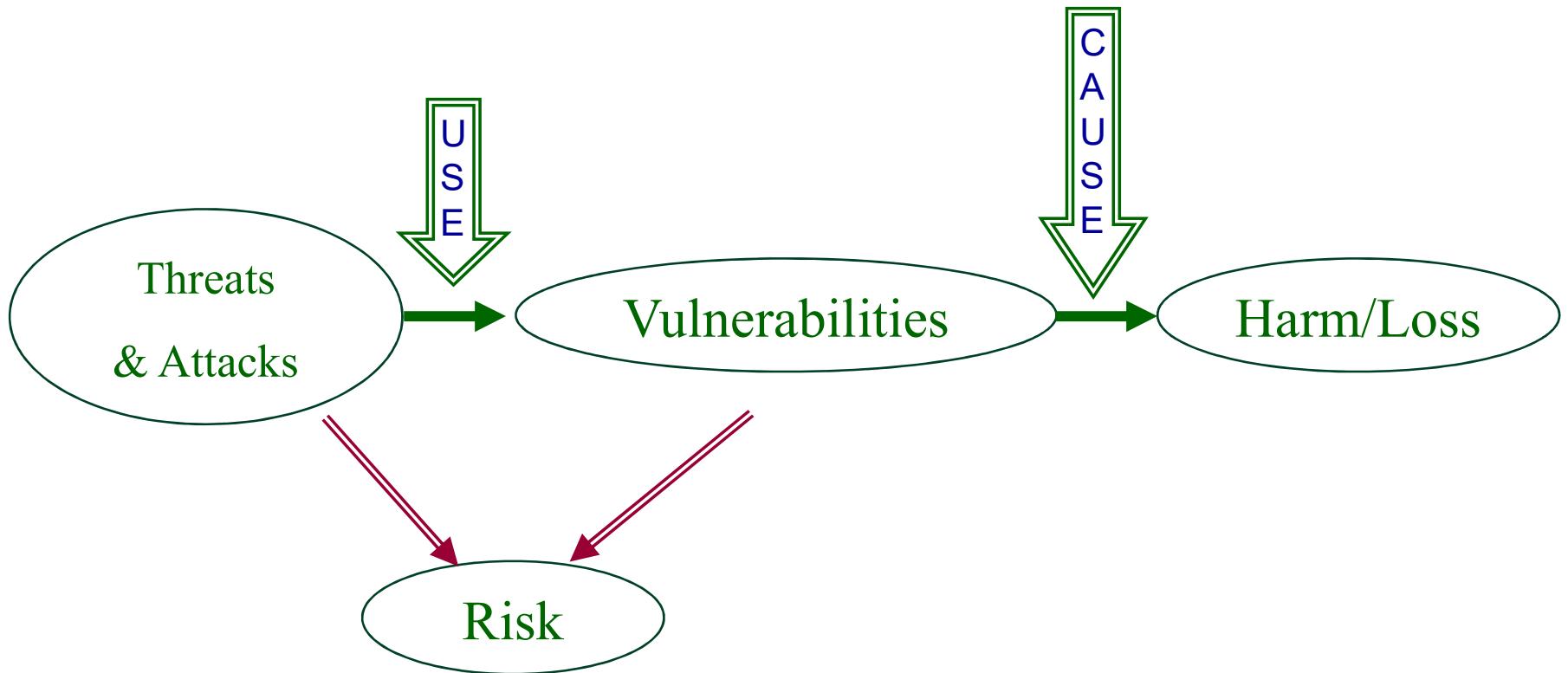
- Defensive Side
- Offensive Side



Offensive and Defensive Operations in IA

- **Defensive Operations**
 - Legal (Security Control)
 - **Offensive Operations**
 - **Illegal (Security Violation)**
 - Unintentional
 - Outsider
 - Insider
 - Intentional
 - Outsider
 - Insider
 - **Legal (Security Testing, a.k.a, Penetration Testing)**
 - Outsider working for insider
 - Insider
 - **Legal & Illegal (Cyber Warfare)**
 - Government/Agency against Government/Agency
-

Offensive Goal



Terms

Threat

- Agent that can inflict harm to an asset or cause security violations

Attack

- Infliction of harm to an asset or causing security violations

Vulnerability

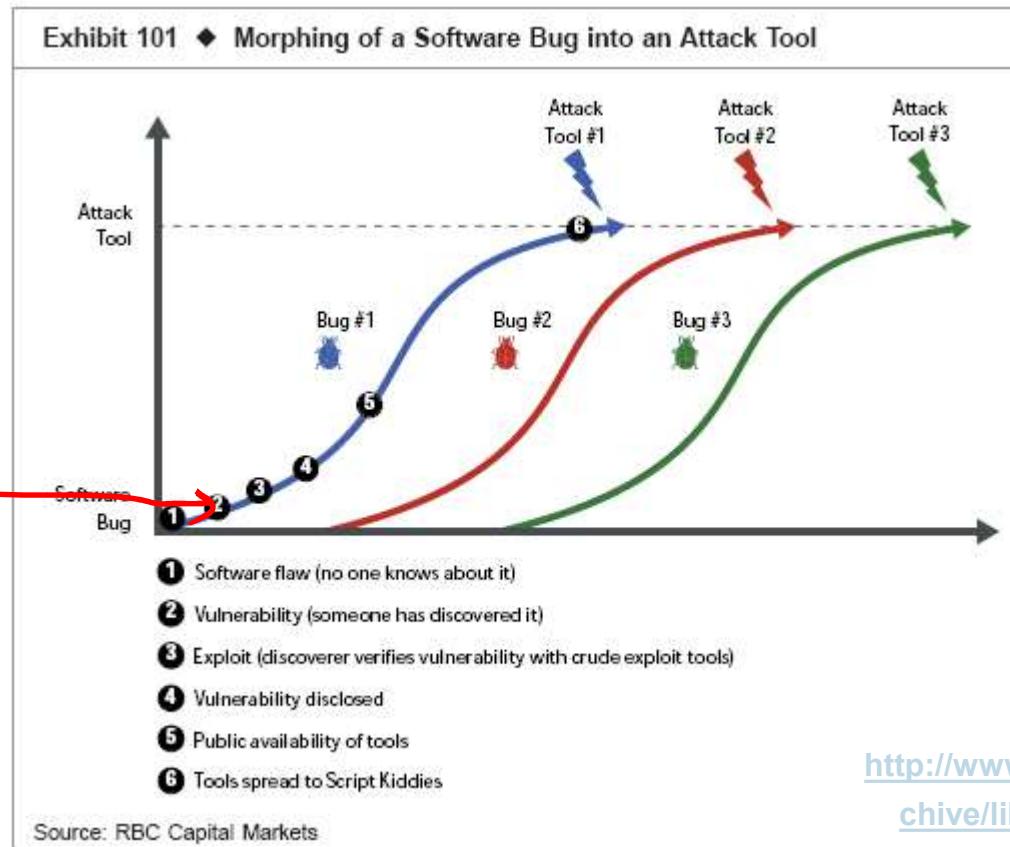
- A weakness in security procedures or system design, implementation, or operation that can be used to cause security policy violation

Risk

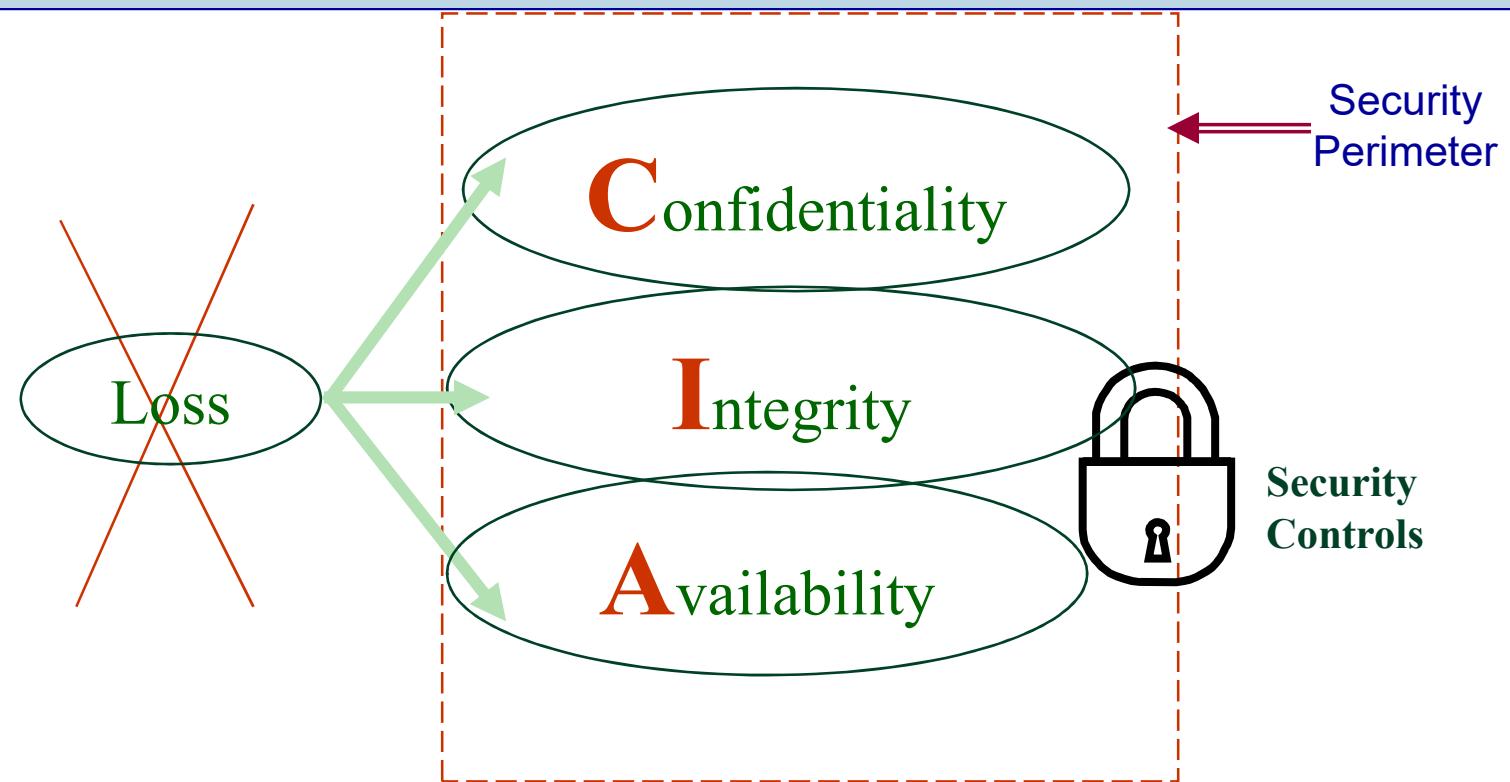
- Likelihood that a particular threat can exploit a particular vulnerability or a set of vulnerabilities to violate security policy
-

From Simple Flaw to Hacker's Tool

'0' DAY
EXPLOITS



Defensive Goal



CIA Goal of IA

- Confidentiality
 - Keeping data and resources hidden
- Integrity (Data and Origin)
 - Keeping data (and data sources) and resources uncorrupted
- Availability
 - Keeping data and resources usable
- ❖ Accountability (a.k.a. Non-Repudiation)
 - Holding one accountable for action

CIA: Confidentiality, Integrity, Availability

- The three pillars of security: the CIA Triad
 - **Confidentiality:** *information and functions can be accessed only by properly authorized parties*
 - **Integrity:** *information and functions can be added, altered, or removed only by authorized persons and means*
 - **Availability:** *systems, functions, and data must be available on-demand according to any agreed-upon parameters regarding levels of service*

Privacy Goal

- Protection and proper handling of sensitive personal information to control the access of others to self
 - Privacy is a right of individuals
 - Confidentiality can relate to individuals, organizations, assets
-

General Classes of Threats

- Disclosure
 - Deception
 - Disruption
 - Usurpation
-

Unauthorized Disclosure

Threat Consequences, and the Types of Threat Actions That Cause Each Consequence

(Based on RFC 2828/4949 Internet Security Glossary)

Threat Consequence	Threat Action (attack)
Unauthorized Disclosure A circumstance or event whereby an entity gains access to data for which the entity is not authorized.	Exposure: Sensitive data are directly released to an unauthorized entity. Interception: An unauthorized entity directly accesses sensitive data traveling between authorized sources and destinations. Inference: A threat action whereby an unauthorized entity indirectly accesses sensitive data (but not necessarily the data contained in the communication) by reasoning from characteristics or byproducts of communications. Intrusion: An unauthorized entity gains access to sensitive data by circumventing a system's security protections.

<http://www.rfc-base.org/txt/rfc-2828.txt>

Slide Courtesy: Mary Ellen Weisskopf, UAB

Unauthorized Deception

Threat Consequences, and the Types of Threat Actions That Cause Each Consequence

(Based on RFC 2828)

Threat Consequence	Threat Action (attack)
Deception A circumstance or event that may result in an authorized entity receiving false data and believing it to be true.	Masquerade: An unauthorized entity gains access to a system or performs a malicious act by posing as an authorized entity. Falsification: False data deceive an authorized entity. Repudiation: An entity deceives another by falsely denying responsibility for an act.

<http://www.rfc-base.org/txt/rfc-2828.txt>

Slide Courtesy: Mary Ellen Weisskopf, UAB

Unauthorized Disruption

Threat Consequences, and the Types of Threat Actions That Cause Each Consequence

(Based on RFC 2828)

Threat Consequence	Threat Action (attack)
Disruption A circumstance or event that interrupts or prevents the correct operation of system services and functions.	Incapacitation: Prevents or interrupts system operation by disabling a system component. Corruption: Undesirably alters system operation by adversely modifying system functions or data. Obstruction: A threat action that interrupts delivery of system services by hindering system operation.

<http://www.rfc-base.org/txt/rfc-2828.txt>

Slide Courtesy: Mary Ellen Weisskopf, UAB

Unauthorized Usurpation

Threat Consequences, and the Types of Threat Actions That Cause Each Consequence

(Based on RFC 2828)

Threat Consequence	Threat Action (attack)
Usurpation A circumstance or event that results in control of system services or functions by an unauthorized entity.	Misappropriation: An entity assumes unauthorized logical or physical control of a system resource. Misuse: Causes a system component to perform a function or service that is detrimental to system security.

<http://www.rfc-base.org/txt/rfc-2828.txt>

Slide Courtesy of: Mary Ellen Weisskopf, UAB

Specific Types of Attacks

- Snooping/Sniffing
 - Spoofing
 - Modification
 - Repudiation of Origin
 - Delay
 - Denial of Receipt
 - Denial of Service
-

Venues for Security Controls

- Hardware**
 - Software**
 - Data**
 - In processing
 - In transit
 - In storage
 - People**
-

Defense in Depth

1) Prevent

- Securing an environment to avoid penetration

2) Deter

- Applying protection mechanisms to hurdle intruder efforts and thus causing delays in achieving a malicious goal

3) Detect

- Ensuring visibility of suspicious activities

4) Response

- Reacting to security incidents by notification, eradication, interdiction, prosecution
- Continuing to survive to some extent

5) Recover

- Assessing and repairing damage
- Improving



Next

- Tuesday: Assignment 2
-

Security Principles

**CSC 4575/5575
Spring 2019**

Slide Source:

Introduction to Computer Security, Matt Bishop, Addison Wesley, Chapter 12

Security Principles

- 1975 Saltzer and Schroeder's fundamental principles of security

 - Benefit:
 - used in design, implementation and/or configuration to prevent loopholes
 - used as the basis of a review checklist

 - Main goal: restriction with simplicity
-

The Principles

- Open design**
 - Fail-safe defaults**
 - Least privilege**
 - Economy of mechanism**
 - Separation of privileges**
 - Complete mediation**
 - Least common mechanism**
 - Psychological acceptability**
-

Least Privilege

- ❑ A user/application/service should be given only those privileges necessary to complete its task
 - *Privilege means permissions determining direct actions on the entity in question*
 - **Function controls assignment**
 - **Minimal set of rights**
 - Rights added as needed, discarded after use

Principle of Least Authority

- ❑ A user/application/service should be given only those authorities as necessary to complete its task
 - Authority means what effects it has on the entity in question either directly or indirectly through another user/application/service*

Fail-Safe Defaults

- Default action is to deny access**
 - Exclude-fail is better than permit-fail
 - Permit as needed
 - If unable to complete task, undo**
-

Economy of Mechanism

- Keep it as simple as possible
 - Simpler means less can go wrong
-

Complete Mediation

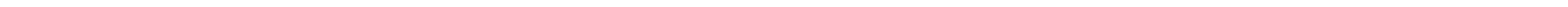
- Check every access
 - Every time
 - No bypass
-

Open Design

- Strength of security should not depend on secrecy of design or implementation (or configuration)
 - Does not apply to information such as passwords or cryptographic keys
-

Separation of Privilege

- Require multiple conditions to grant privilege/access**
 - Separation of duty



Least Common Mechanism

- Mechanisms/Resources should not be shared**
 - **Information can flow along shared channels**
-

Psychological Acceptability

- Security mechanisms should not add to difficulty of accessing resource
 - Hide complexity introduced by security mechanisms
 - Ease of installation, configuration, use

- Principle of Least Astonishment
 - Mechanisms should designed so that users understand the reason the mechanism works the way it does and its simple to understand
 - The result of performing some operation should be obvious, consistent, and predictable....

<http://c2.com/cgi/wiki?PrincipleOfLeastAstonishment>

The Principles

- Open design ➤ No secrecy
 - Fail-safe defaults ➤ No default permit
 - Least privilege ➤ No more privilege than needed
 - Economy of mechanism ➤ No complexity
 - Separation of privileges ➤ No single responsibility
 - Complete mediation ➤ No bypass
 - Least common mechanism ➤ No sharing
 - Psychological acceptability ➤ No hardship/surprises
-

Interesting Reads

- <https://buildsecurityin.us-cert.gov/articles/knowledge/principles/design-principles>
 - <http://emergentchaos.com/the-security-principles-of-saltzer-and-schroeder>
-

Checksum

- Digital checksums are compact representation of data
 - Provides integrity service
- Simple checksum uses 1 unit of data to detect errors in rest
 - Use of ASCII parity bit
 - ASCII has 7 bits; 8th bit is “parity” (can be 0/1)
 - Types
 - Even parity: parity bit set to get even number of 1 bits
 - Odd parity: parity bit set to get odd number of 1 bits
 - Bob receives “10111101” as bits.
 - If sender is using even parity
 - character was received correctly (Six 1 bits)
 - If sender is using odd parity
 - character was received incorrectly (Six 1 bits)

PPR
B

Cryptographic Checksum/Hash

- Function/method to generate a set of k bits from a set of n bits (where $k \leq n$).
 - For any given message, easy to compute the hash (not vice versa)
 - For a given message, the corresponding hash can not be predicted (only computed to be seen)
 - Same message will produce the same hash
 - Any change in message will generate completely different hash
 - A.k.a.,
 - Message digest
 - Message Integrity Code (MIC)
 - Modification Detection Code (MDC)
-

Hash is NOT Encryption

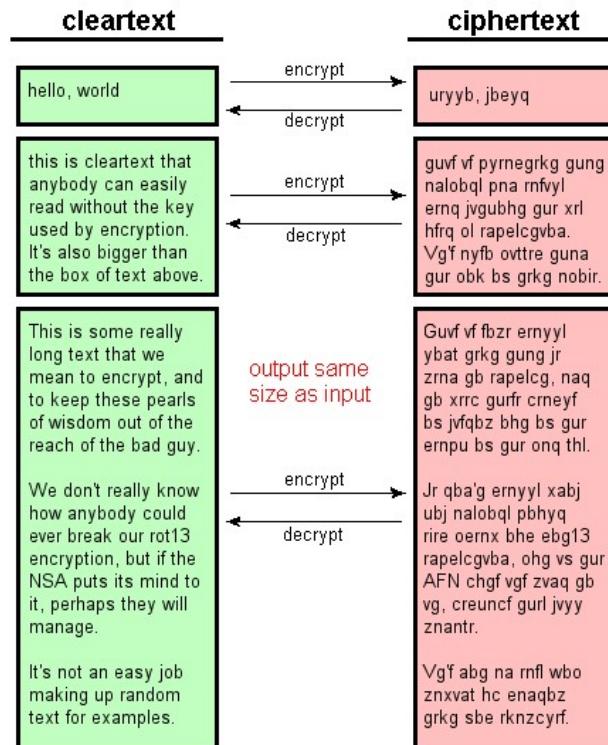


Fig. 1: Encryption - a two-way operation

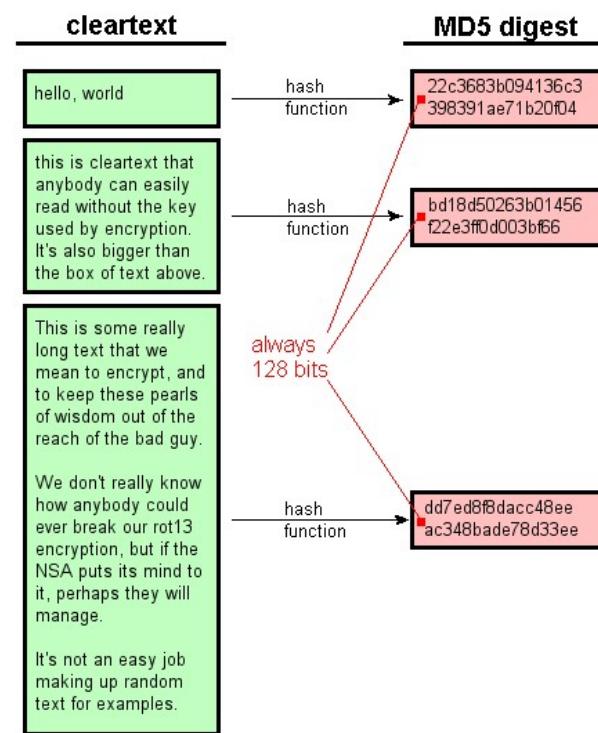


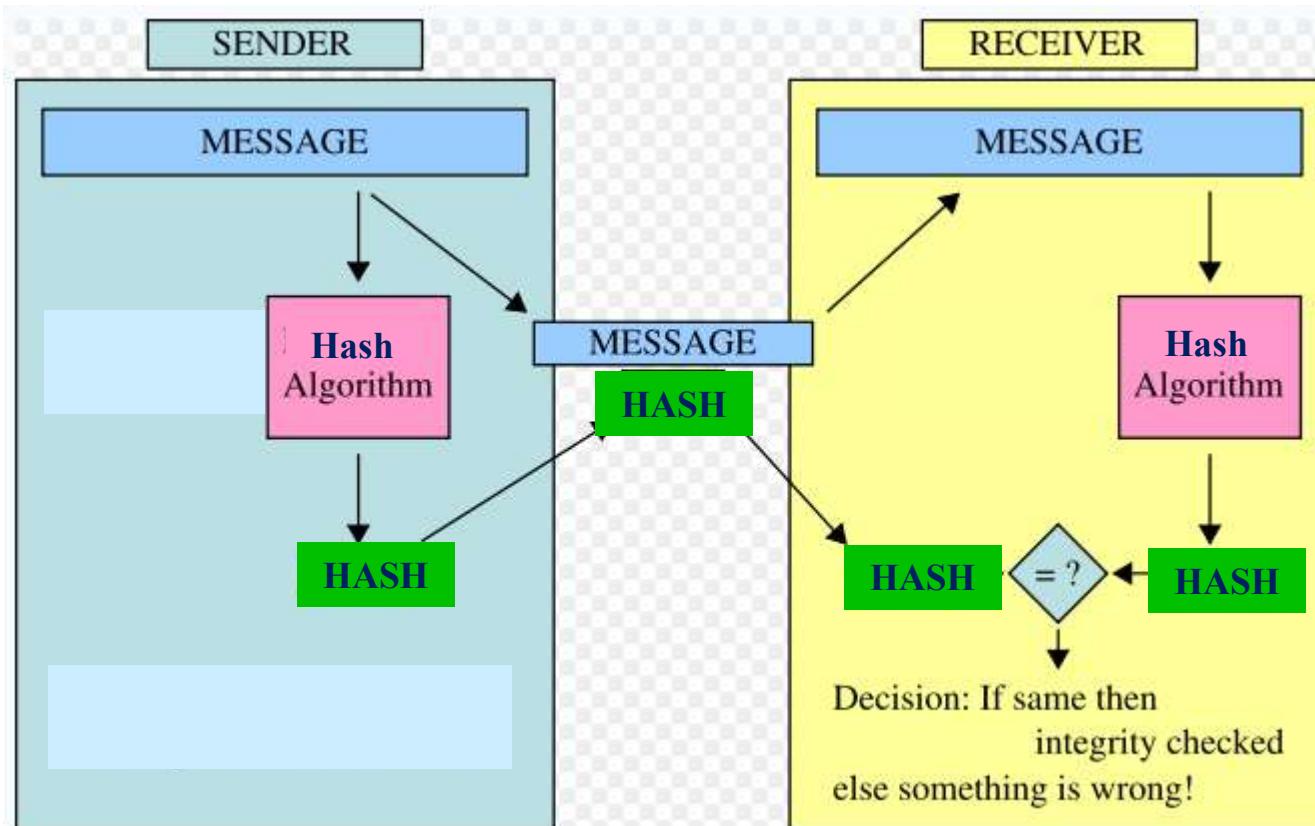
Fig. 2: Hashing - a one-way operation

<http://www.unixwiz.net/techtips/iguide-crypto-hashes.html>

Cryptographic Hash Properties

- Impossible to derive original message from digest
 - Fixed-length digest regardless of message size
 - A digest should be result of the whole message, not a portion of it
-

Checksum/Hash

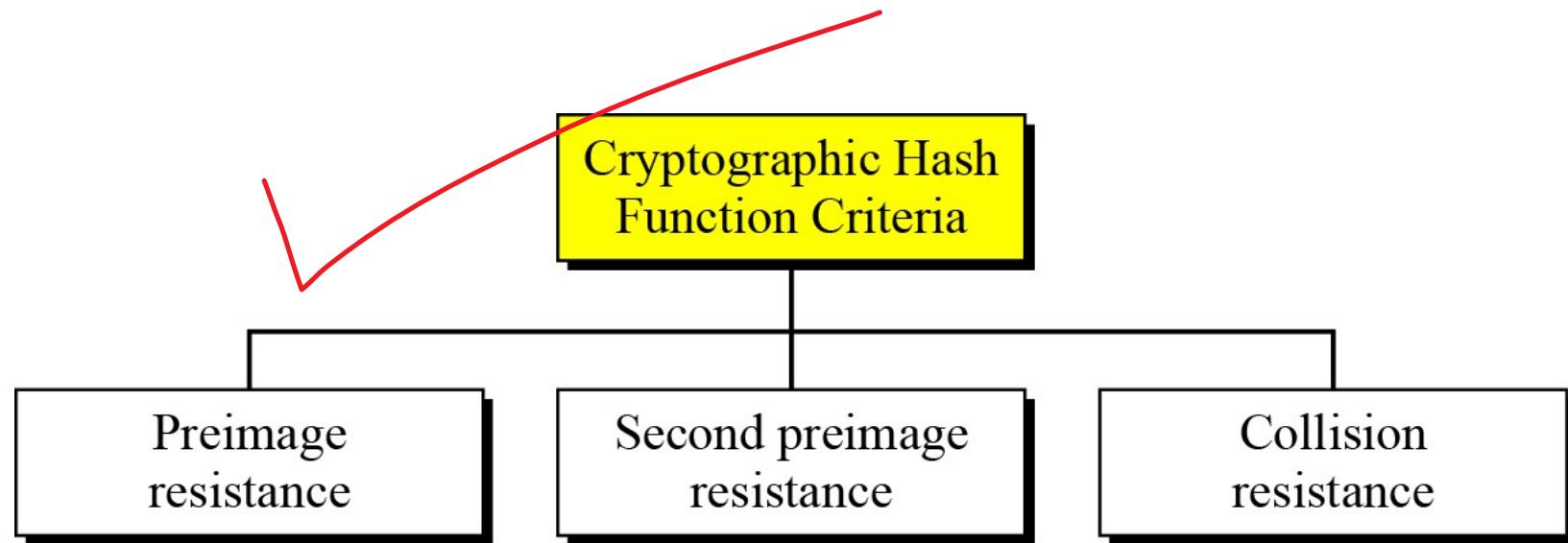


Adapted from <http://en.wikipedia.org/wiki/Image:MAC.svg>

Applications of Hash

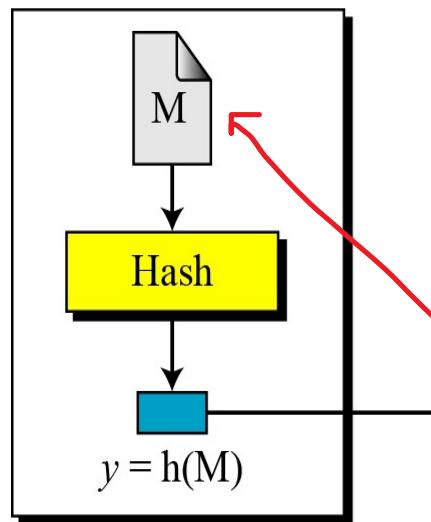
- Message (in communication) integrity
 - Document/file (in storage) integrity
 - Application/system state integrity
 - Identifier of entity
 - Key generation
-

Resistance Criteria of Cryptographic Hash Function

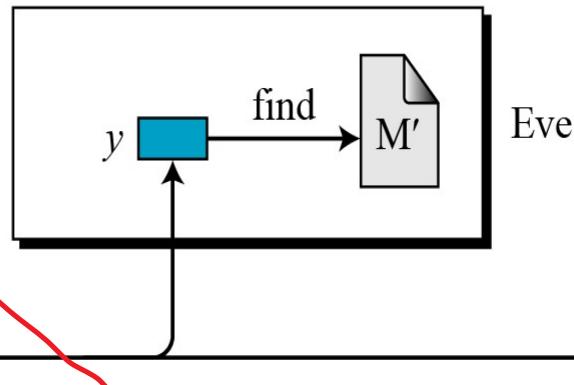


Preimage Resistance

M: Message
Hash: Hash function
 $h(M)$: Digest

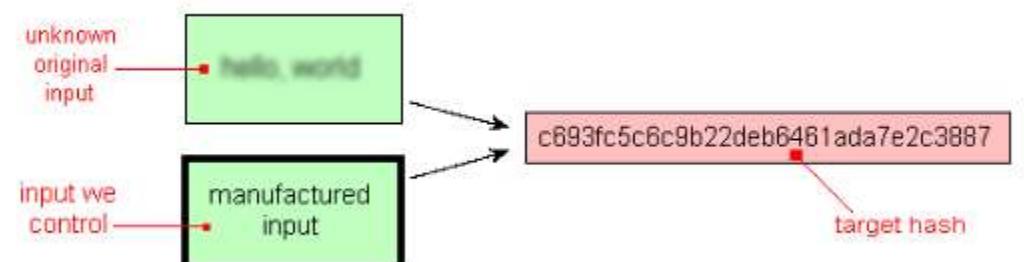


Given: y
Find: any M' such that
 $y = h(M')$

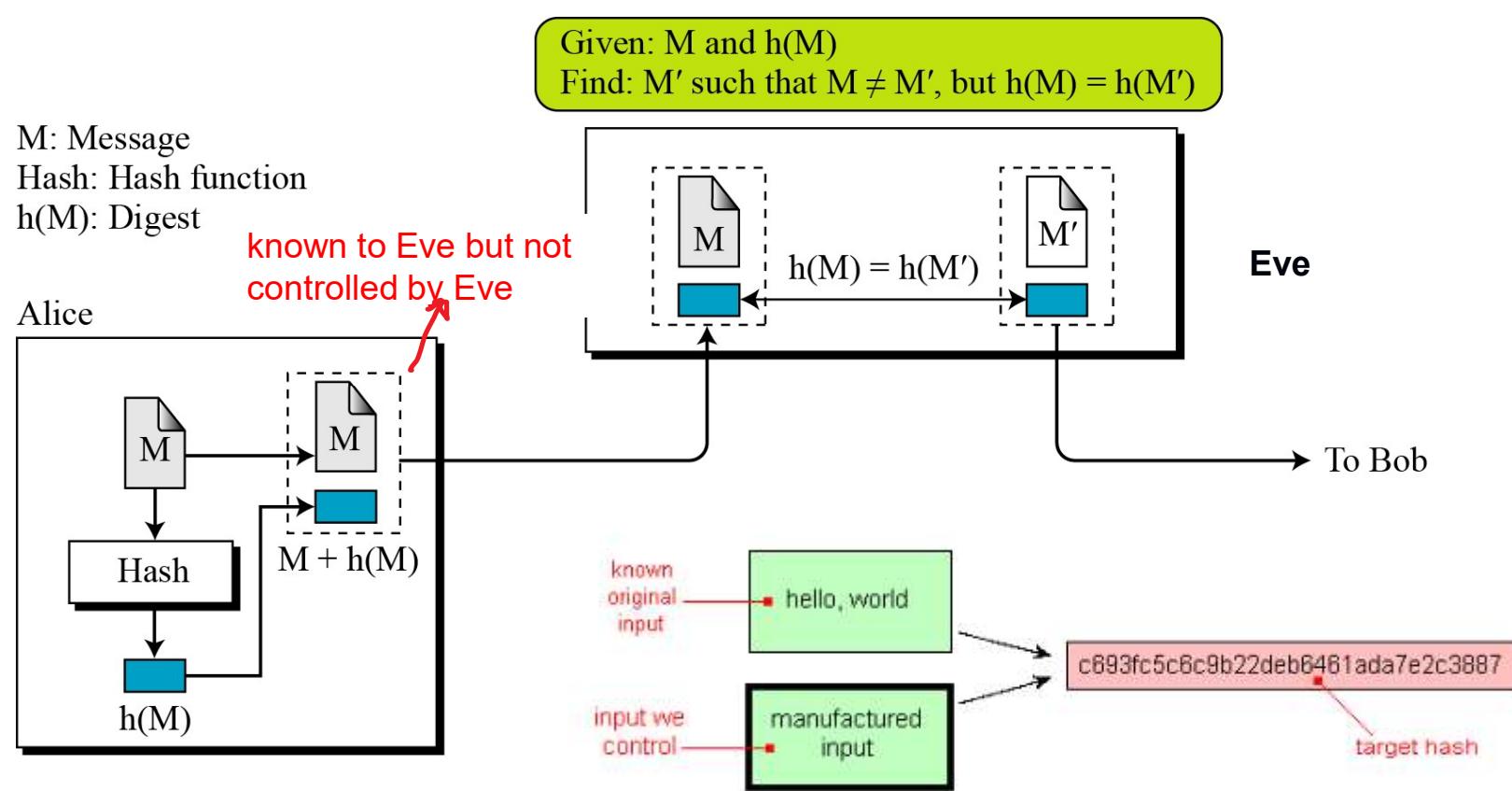


To Bob

unknown to Eve



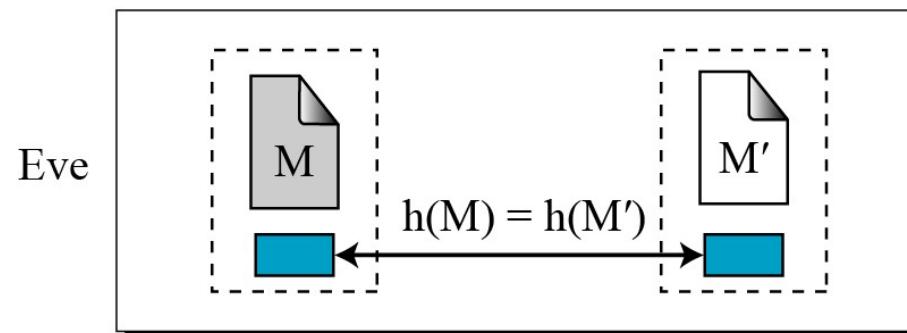
Second Preimage



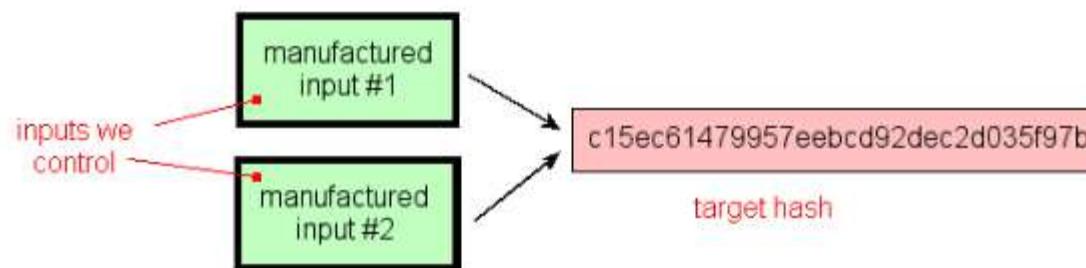
Collision Resistance

M: Message
Hash: Hash function
h(M): Digest

Find: M and M' such that $M \neq M'$, but $h(M) = h(M')$



Both controlled by Eve



Summary of Hash Resistance

- No other message should produce the same digest
 - Given only hash/digest PR
 - Given only hash/digest and original message SR
- No two messages should result in the same hash/digest CR

Collisions

- ☐ When two different messages have same hash, they are called collisions.

If $x \neq x'$ and $h(x) = h(x')$, x and x' are a *collision*

- Pigeonhole principle: if there are n containers for $n+1$ objects, then at least one container will have 2 objects in it.
 - Example: if there are 32 files and 8 possible cryptographic checksum values, at least one value corresponds to at least 4 files

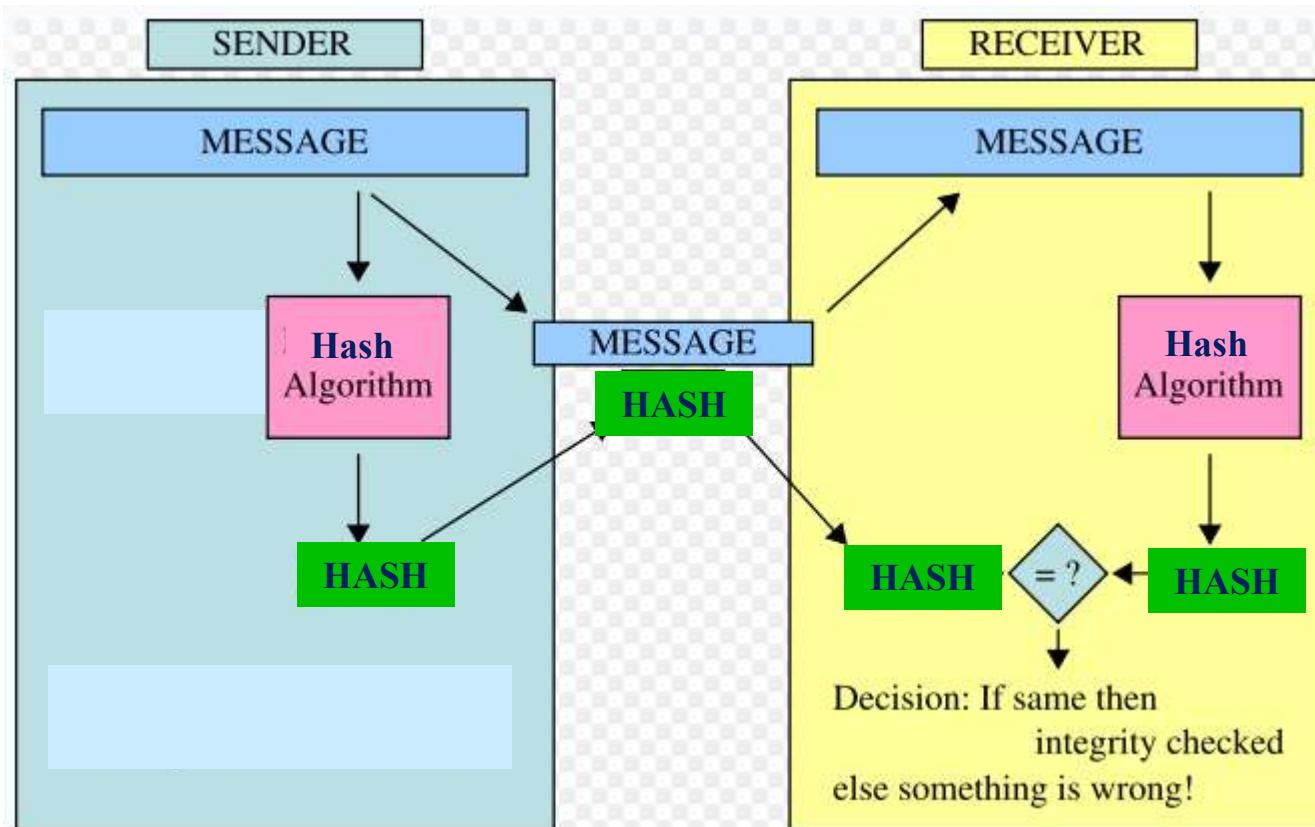
- ☐ Birthday Attack

- Used for finding collisions to commit forgery
- Possible because of birthday paradox
 - Probability that from a group of random number of people at least a pair will have the same birthday
 - The probability increases with pigeonhole principle, for example, this probability will be maximum when there is more than 366 people in the group

Key use in checksums

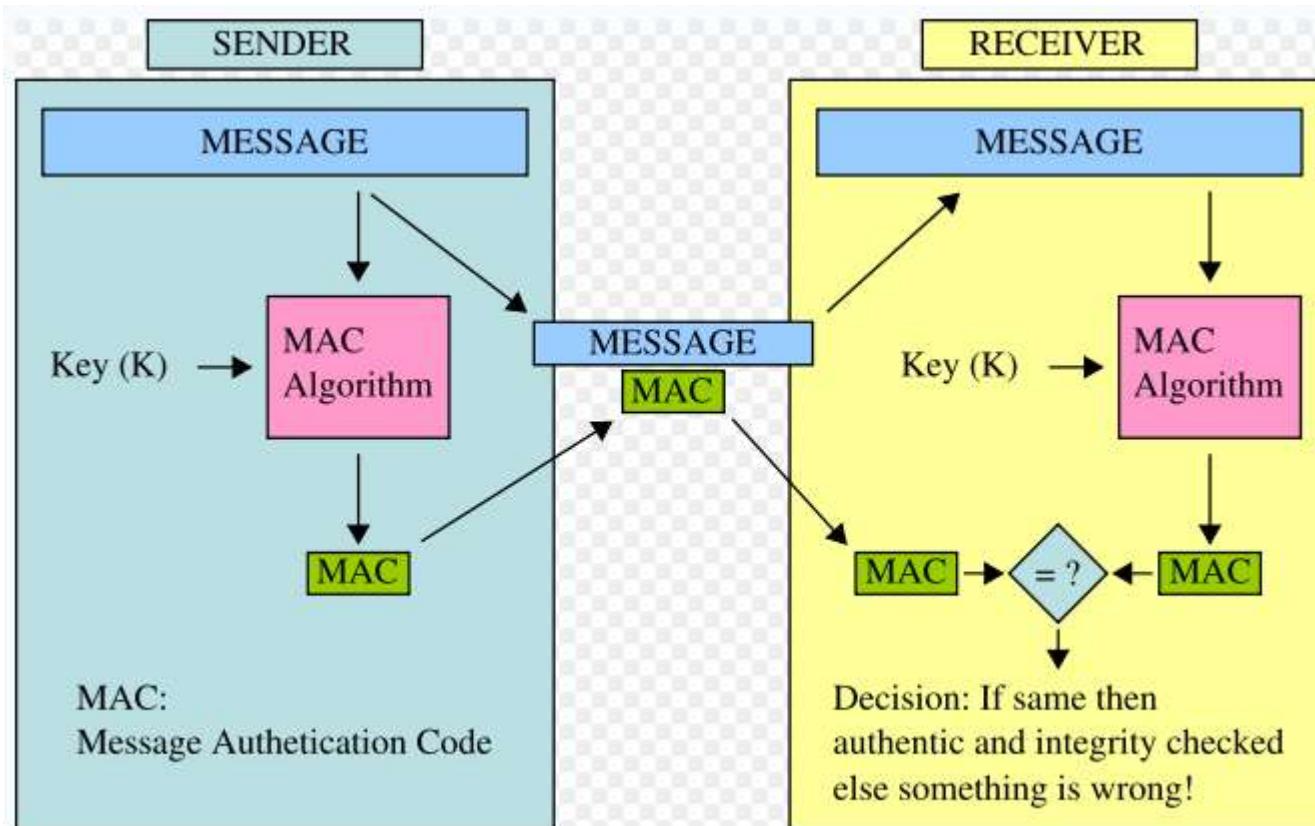
- ❑ Keyless cryptographic checksum: requires no cryptographic key
 - MD5 and SHA-1 are best known; others include MD4, HAVAL, and Snefru
 - Susceptible to forgery
- ❑ Keyed cryptographic checksum: requires cryptographic key
 - Can be used for message authentication (origin integrity)
 - Called Message Authentication Code (MAC)
 - Not same as Digital Signatures
- ❑ Keyed checksum provides more assurance than keyless

Checksum/Hash



Adapted from <http://en.wikipedia.org/wiki/Image:MAC.svg>

MAC



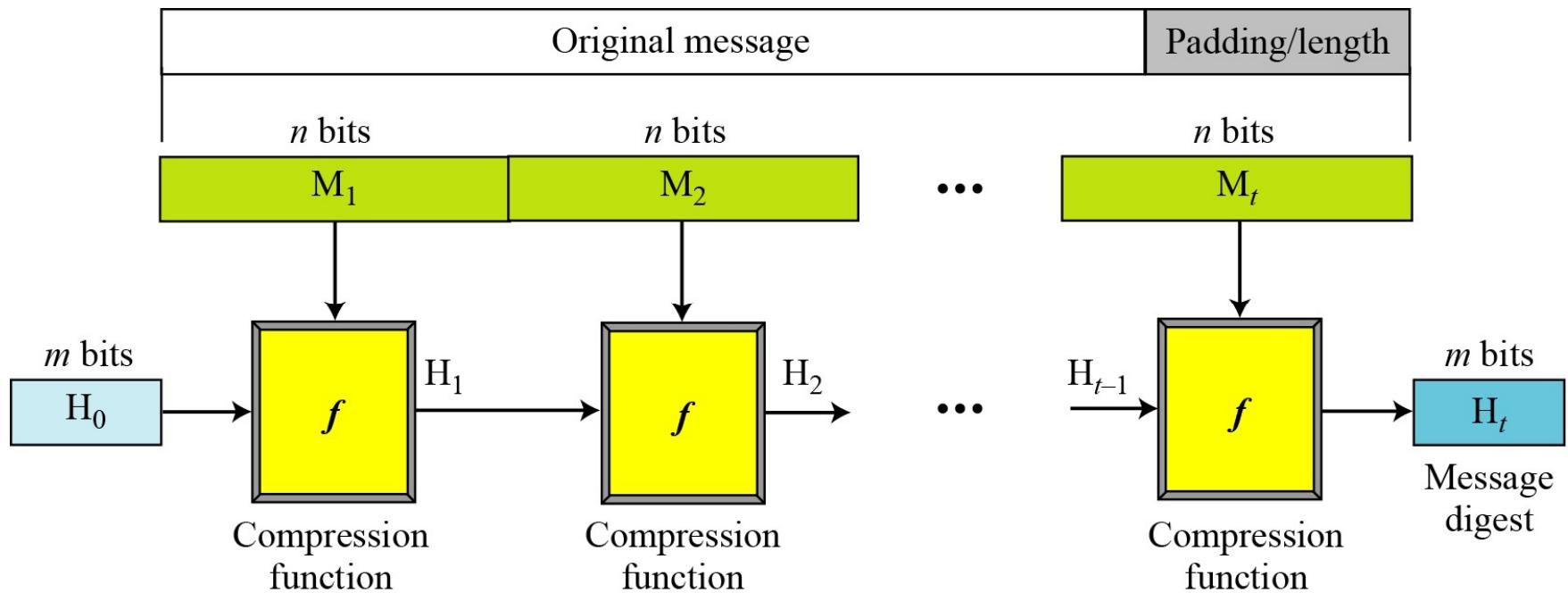
<http://en.wikipedia.org/wiki/Image:MAC.svg>

Use of Key in MAC

- Usually the key is concatenated with the message and then hashed to produce MAC
 - Prefix, postfix or both
 - Subject to brute force attack for key, unless size of key makes it difficult
 - Strength of MAC depends on strength of hash algorithm used
 - To improve security, iterated or nested MACs are used
-

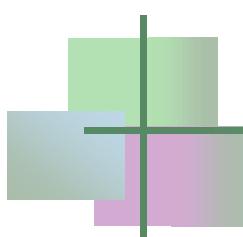
Iterated Hash with Compression Function

Merkle-Damgård Scheme



Compression (Hash) Functions

- Generates an m-bit string from a string of n-bit where $m \leq n$
- Two types
 - Made from Scratch
 - MD5, SHA
 - Based on block ciphers
 - Whirlpool



SHA Features

Table 12.1 Characteristics of Secure Hash Algorithms (SHAs)

Characteristics	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Maximum Message size	$2^{64} - 1$	$2^{64} - 1$	$2^{64} - 1$	$2^{128} - 1$	$2^{128} - 1$
Block size	512	512	512	1024	1024
Message digest size	160	224	256	384	512
Number of rounds	80	64	64	80	80
Word size	32	32	32	64	64

~~CBC-MAC or CMAC~~ ✓(Hash with block cipher)

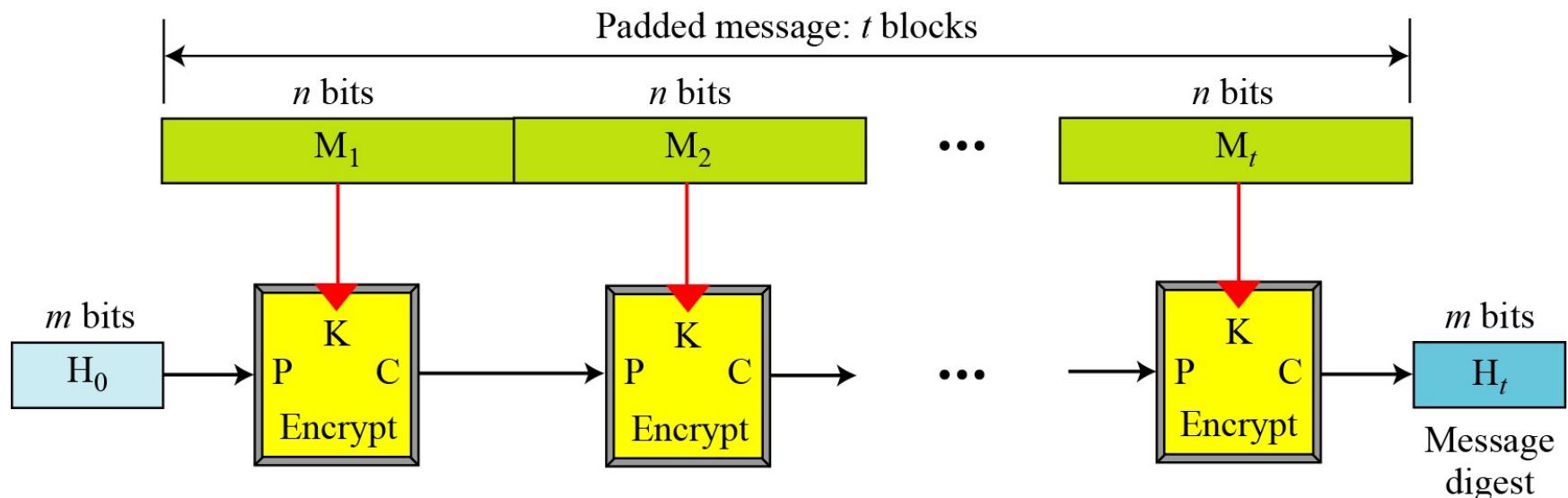
□ DES in CBC mode

- Encipher message, use last n bits as checksum.
 - Requires a key to encipher, so it is a keyed cryptographic checksum.

INTEGRITY
WITH
DES

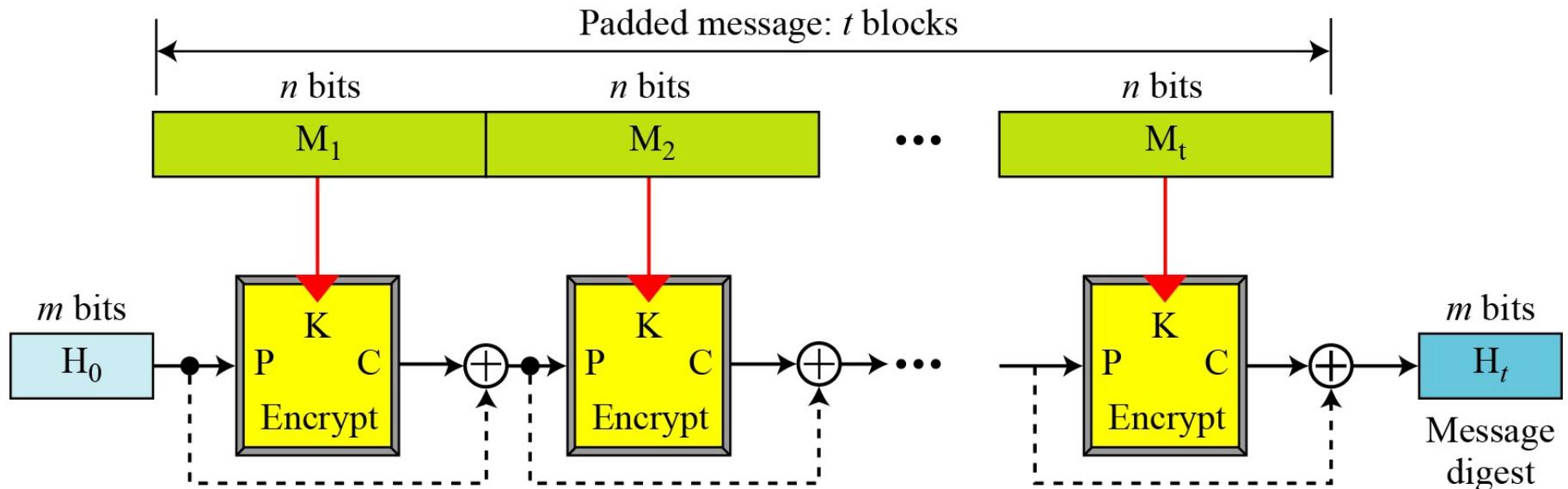
Iterated Hash with Block Ciphers

Rabin scheme



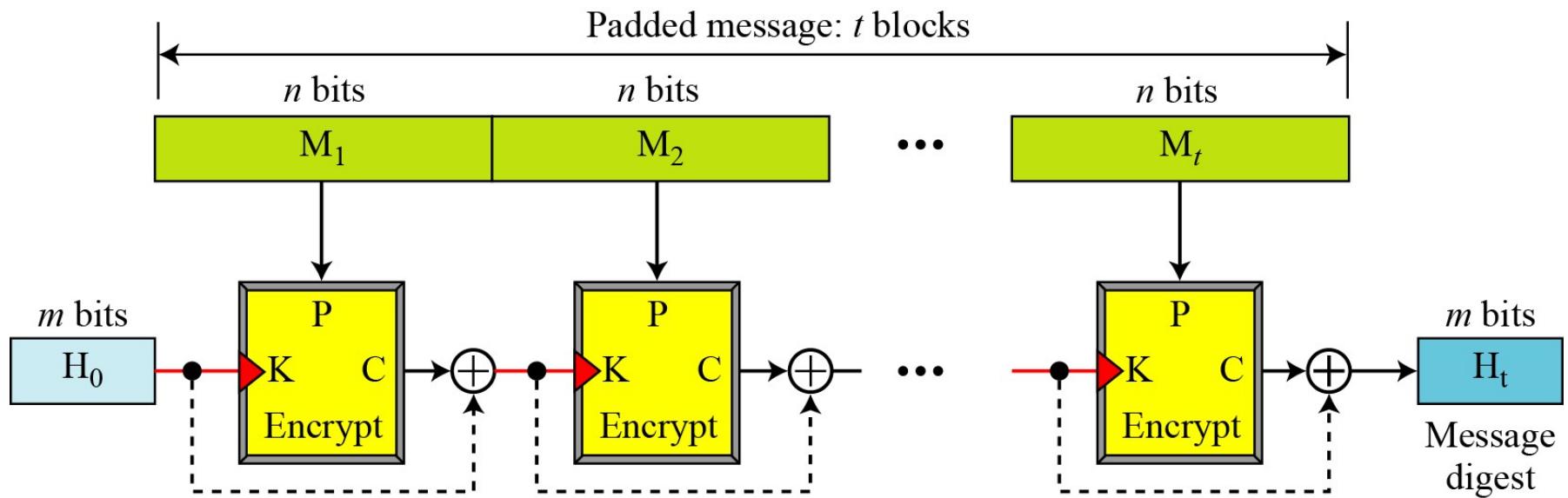
Iterated Hash with Block Ciphers

Davies-Meyer scheme



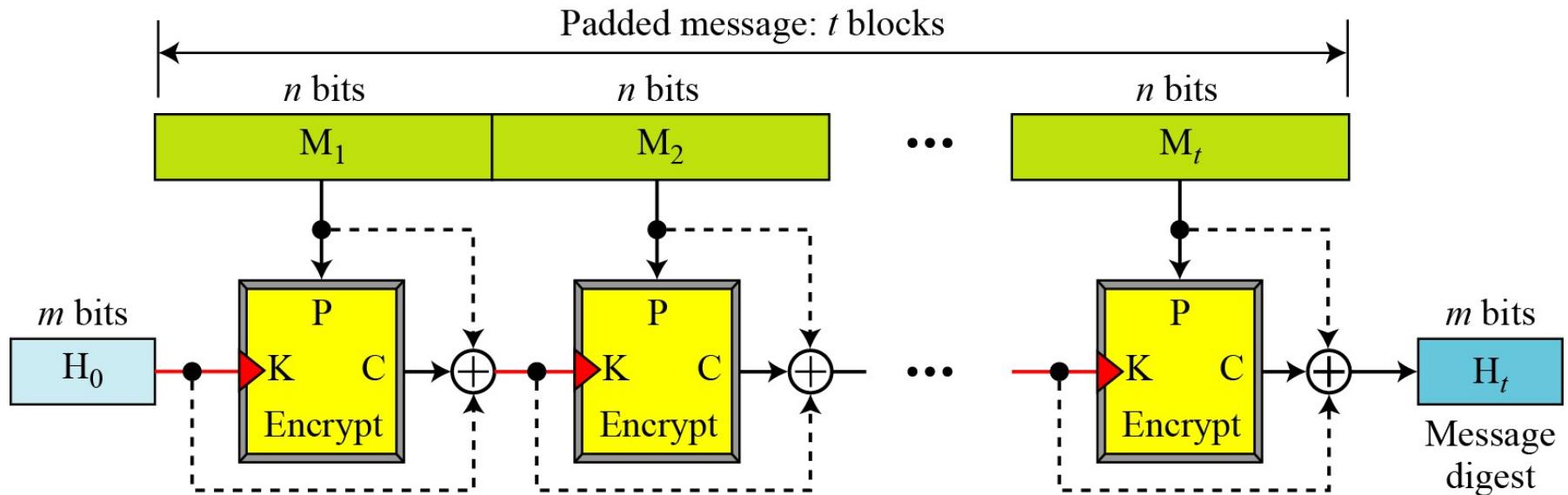
Iterated Hash with Block Ciphers

Matyas-Meyer-Oseas scheme



Iterated Hash with Block Ciphers

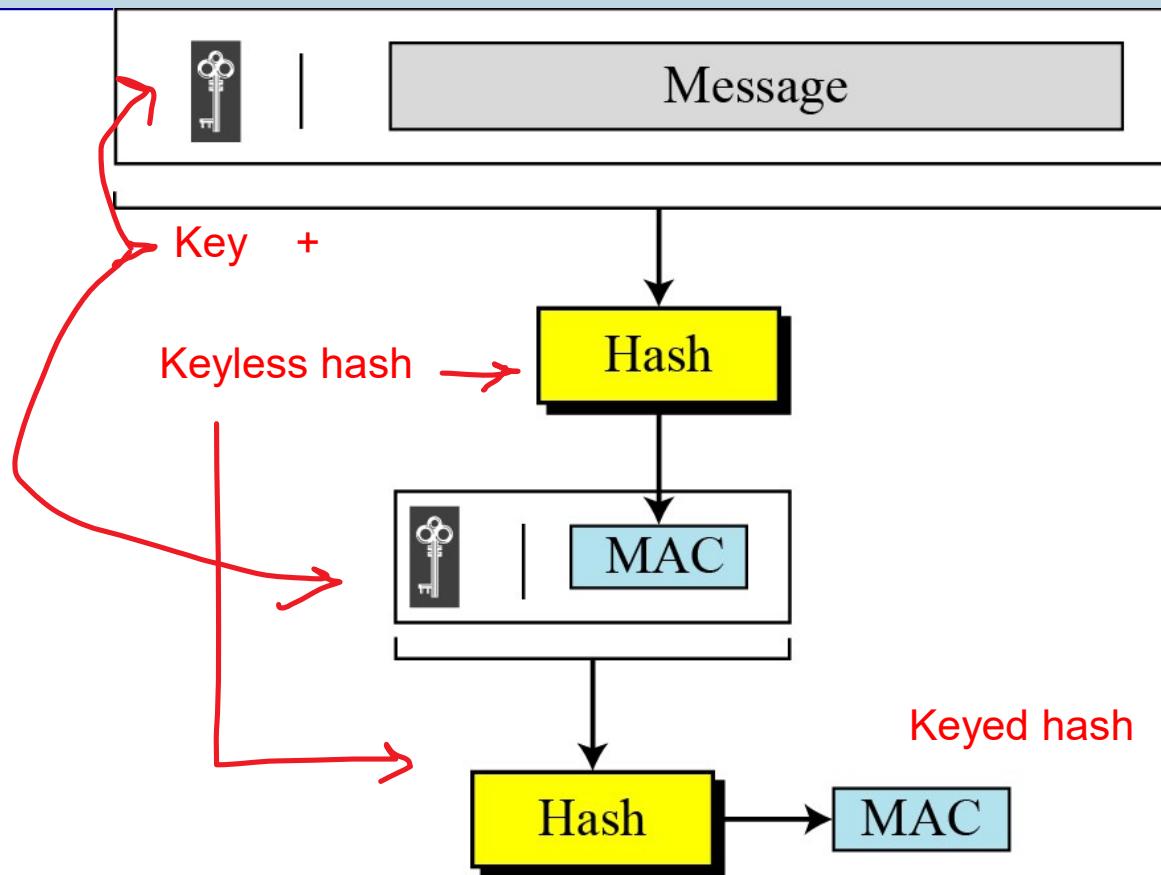
Miyaguchi-Preneel scheme



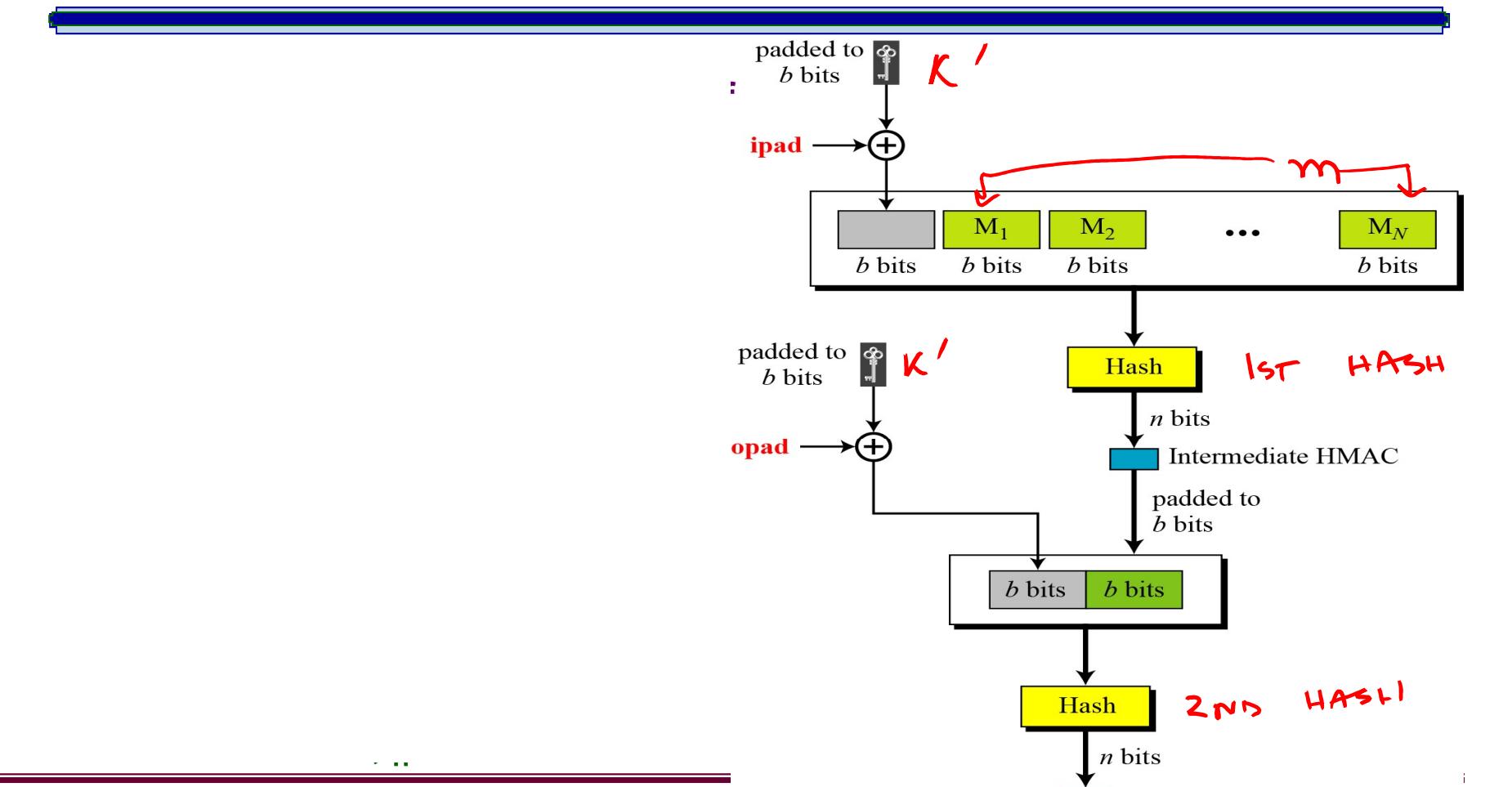
Nested Hash or HashMAC (HMAC)

- Purpose**
 - Cryptographic algorithms to produce keyed checksums had restrictions on import/export in many countries BUT keyless checksums don't <https://www.emc.com/emc-plus/rsa-labs/standards-initiatives/united-states-cryptography-export-import.htm>
 - So, idea is to use key with keyless checksums to produce keyed checksums
- General name for algorithm to produce keyed checksums (MAC) from keyless checksums (MIC)**
 - Example, if hash function is MD5, then the resulting HMAC algorithm is termed HMAC-MD5
- Strength of the HMAC primarily depends upon**
 - cryptographic strength of the underlying hash function
 - size and quality of the key
- Not affected by collision problem.**
- Uses nested checksums.**

Nested Hash Concept



HMAC



Digital Signature

- ❑ Construct that authenticates origin of message in a manner provable to a disinterested third party (“judge”)
 - Serves as evidence
- ❑ Sender cannot deny having sent message
 - Provides “non-repudiation” service
 - One could claim the cryptographic key was stolen or compromised
- ❑ Different from MAC
 - MAC do not offer non-repudiation

Example

- Classical: Alice, Bob share key k

- Alice sends $m \parallel \{m\}k$ to Bob

Is this a digital signature?

No.

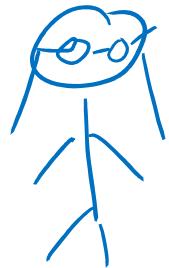
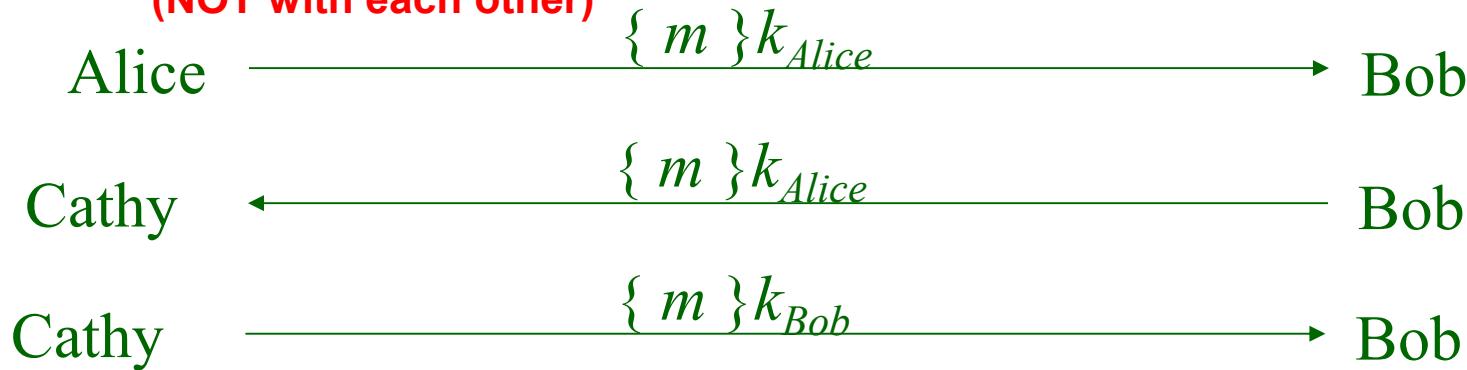
Third party cannot determine whether Alice or Bob generated message.

It is rather a MAC that supports origin integrity.

Non-Repudiation with Classical/Symmetric Cryptography

- ☐ Intervention of trusted third party is required to achieve non-repudiation with classical cryptography.

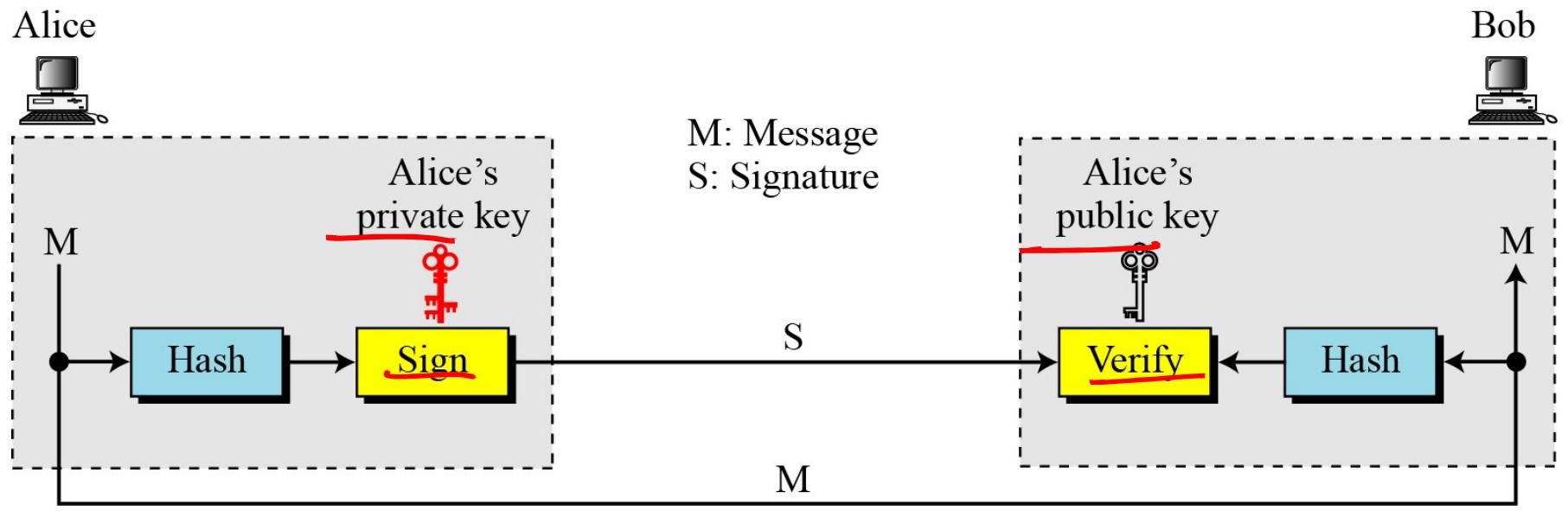
- Alice, Bob each share keys with TRUSTED 3RD PARTY Cathy (NOT with each other)



- ☐ To resolve dispute, judge gets $\{ m \} k_{Alice}$ and $\{ m \} k_{Bob}$, from Bob and Alice respectively, and has Cathy decipher them to check for forgery

Public Key Digital Signatures

- Enciphering message (or hash) with private key produces digital signature



Example

- Alice's keys are d_{Alice} , e_{Alice}
- Alice sends Bob

$$m \parallel \{ m \} d_{Alice}$$

- In case of dispute, judge computes

$$\{ \{ m \} d_{Alice} \} e_{Alice}$$

- and if it is m , Alice signed message
 - She's the only one who knows d_{Alice} !

Forgery and Precaution

- Never sign random documents
- Only signing does not help
- Should be both signed (with sender's private key) and enciphered (with receiver's public key)

- DON'T encipher and then sign
 - Sign first and then encipher

SENDER's PRIVATE

RECEIVER's PUBLIC

DOUBLE
ENCIPHERMENT

Key Differences

□ Differences between MAC and MIC

- MAC used for authentication, MIC used for integrity
- MAC requires key, MIC doesn't
 - Same algorithm on same content= Same MIC
 - Same algorithm on same content w/different key= Different MAC

□ Difference between MAC and Digital Signature

- MAC does not provide non-repudiation like digital signatures because of use of symmetric encryption

□ None provide confidentiality (if used in typical way)

Key Points

- Two main types of cryptosystems: classical and public key
 - Classical cryptosystems encipher and decipher using the same key
 - Public key cryptosystems encipher and decipher using different keys
 - Cryptographic checksums provide a check on integrity
 - Digital signatures provide integrity of origin and content
-

Links of Interest

- <http://kathrynnneugent.com/des.html>
 - <http://www.cs.bham.ac.uk/research/projects/lems/DES/index.jsp>
 - <http://www.unixwiz.net/techtips/iguide-crypto-hashes.html>
-

Security Policies and Models

CSC 4575/5575
Information Assurance and Cryptography
Spring 2019

Sources:

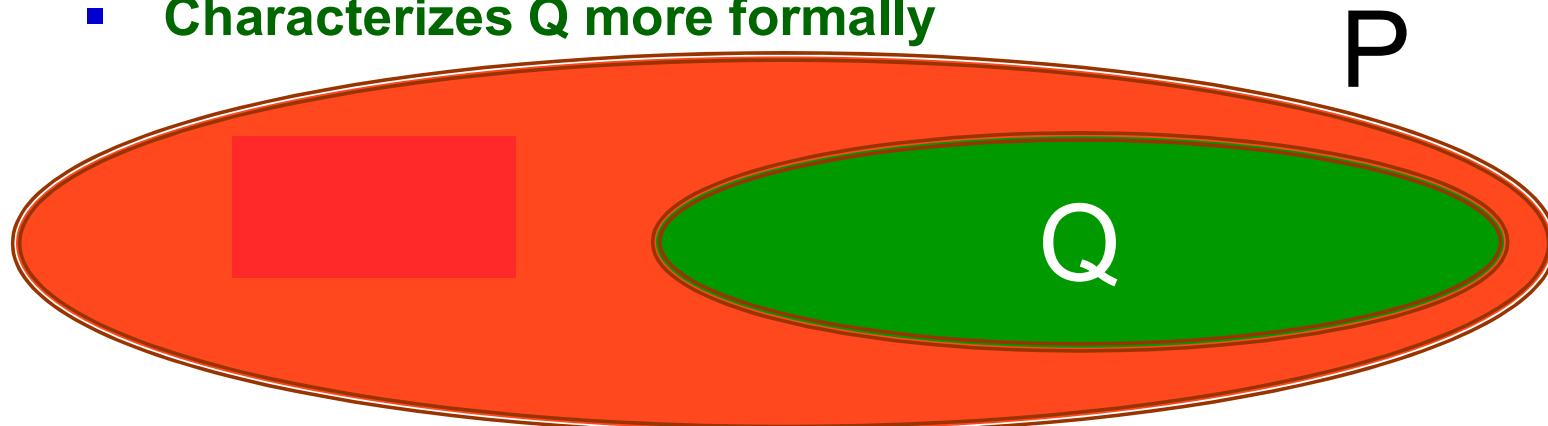
Computer Security: Art and Science, Matt Bishop, Addison Wesley, 2003 [Chapters 1, 2, 4]
Security in Computing, Pfleeger and Pfleeger, Prentice Hall, 2003

Policy, Model & Mechanism

- Security Policy
 - Statement of what is (and not) allowed
 - Security Model
 - Representation of policy
 - Security Mechanism
 - Methods, tools and procedures to enforce policy by implementing model
- MECHANISM
↓ IMPLEMENTS
MODEL
↓ FORMALIZES
POLICY

More on Policy, Model & Mechanism

- Security Policy**
 - Identifies Q within P
- Security Mechanism**
 - Ensures Q and prevents P - Q
- Security Model**
 - Characterizes Q more formally



Protection State (P): System states relevant to protection
Authorized State (Q): System states authorized to reside

Example

- Password protected Eagle Online account — MECHANISM
 - Only active students can access Eagle Online — POLICY
 - If (and only if),
 x is a subset of all students X in Tennessee Tech representing active students,
i.e., $x \in X$, then x can have access to Eagle Online. — MODEL
-

Security Policy

- The set of laws, rules, and practices that regulate how an organization manages, protects, and distributes sensitive information.
- Types
 - Military
 - Commercial

Military/Government Policies

- Focus on confidentiality
- Expressed with Security labels/flags
 - Classification of objects & Clearance of subjects
 - combination of hierarchical sensitivity and non-hierarchical compartments
- Access is controlled by security labels/flags, “need to know” rules

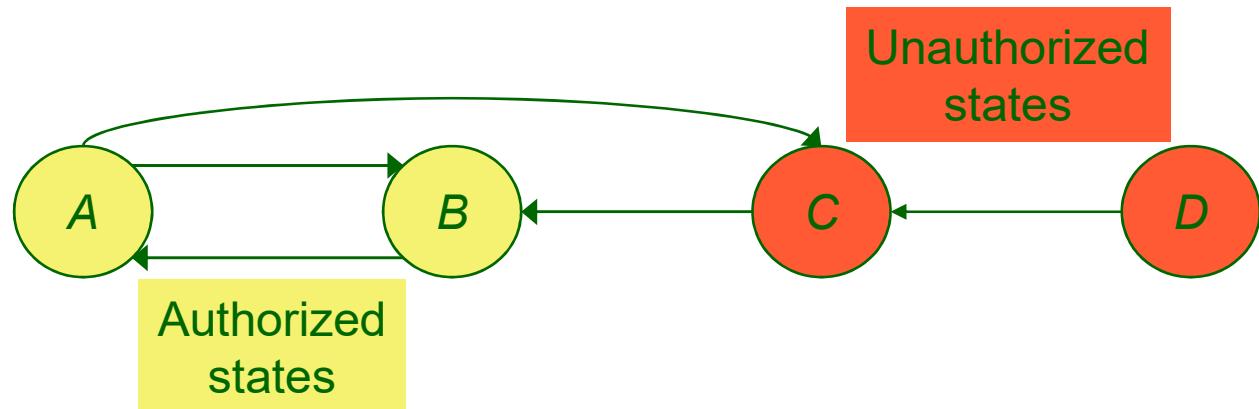
Commercial Policies

- Focus on integrity, availability and non-hierarchical confidentiality

Another look at Security Policy

- From behavioral perspective, if we consider computer systems as finite-state machines with**
 - **Set of states**
 - **Set of transition functions that change state**
- then we can define security policy as**
 - **one which partitions system states into**
 - **Authorized states**
 - **Unauthorized states**

Finite State Machine Example



- Secure System
 - Starts in authorized state
 - Never enters unauthorized state

Confidentiality Property

- X set of entities, I information
- I has *confidentiality* property with respect to X
 - if no $x \in X$ can obtain information from I

~~Confidentiality Policy~~

- About secrecy of information**
 - Does not care about trust
- Confidentiality levels designate extent of secrecy**
- Deals with**
 - Prevention of
 - unauthorized rights
 - Transfer of information without transfer of rights
(information flow)
 - Temporal context (dynamic change of rights)
- Highly developed in Military/Government**

Integrity Property

- X set of entities, I information
- I has *integrity* property with respect to X
 - if all $x \in X$ trust information in I
- Types of *integrity*:
 - trust I itself, its conveyance and storage (**data integrity**)
 - trust I , where I is information about origin of something or an identity (**origin integrity**, authentication)
 - trust I , where I is resource: means resource functions correctly

Integrity Policy

- About trustworthiness of information**
 - Does not care about secrecy
- Integrity levels designate levels of trustworthiness**
- Deals with**
 - Entities that are allowed to alter data
 - Conditions under which data can be altered
 - Limits to change of data
- Highly developed in commercial world**

Availability Property

- X set of entities, I data/resource
- I has *availability* property with respect to X
 - if all $x \in X$ can access I
- ~~Types of availability:~~
 - traditional: x gets access or not (binary)
 - quality of service: x gets a level of access (graded)

Availability Policy

- Deals with
 - What services must be provided
 - To what extent
- Mostly developed in commercial world

Enforcing Policy

- Explicit Policy**
 - X cannot view Y's notes
 - Implicit Policy**
 - Y have to protect notes
-
- If X cheats, only X can be held accountable
 - If X cheats, both X and Y can be held accountable
-
- Security policy must be as explicit as possible
 - Security model must be unambiguous and explicit

Trust Relationships

- Trust and assumption play crucial role in policy, especially, integrity policy
 - Trust is based on assumptions
 - Attackers look for assumptions and trusted users to find possible weak points in implementation of policy

Role of Trust

□ Higher level assumption example

- Administrator installs patch with the trust that
 - Patch came from vendor
 - Not tampered with in transit
 - Vendor tested patch thoroughly
 - Vendor's test environment corresponds to local environment
 - Patch is installed correctly

Role of Trust cont.

- Lower level assumption example
 - A security-related program S is formally verified to work with operating system O
 - Proof has no errors
 - » Bugs in automated theorem provers
 - Preconditions hold in environment in which S is to be used
 - S transformed into executable whose actions follow source code
 - » Compiler bugs, linker/loader/library problems
 - Hardware executes the program as intended
 - » Hardware bugs



“Secure” vs. “Trusted”

Secure System

- A Goal
- Asserted based on features
- Either ... or

Trusted System

- A Characteristic
- Judged based on evidence/analysis
- Degrees of Trustworthiness

Security Model

□ Primary purposes

- Provide a framework to aid in understanding
- Provide an unambiguous, often formal, representation of a general security policy

□ Focuses on

- Points of interest in policies
- Specific characteristics of policies

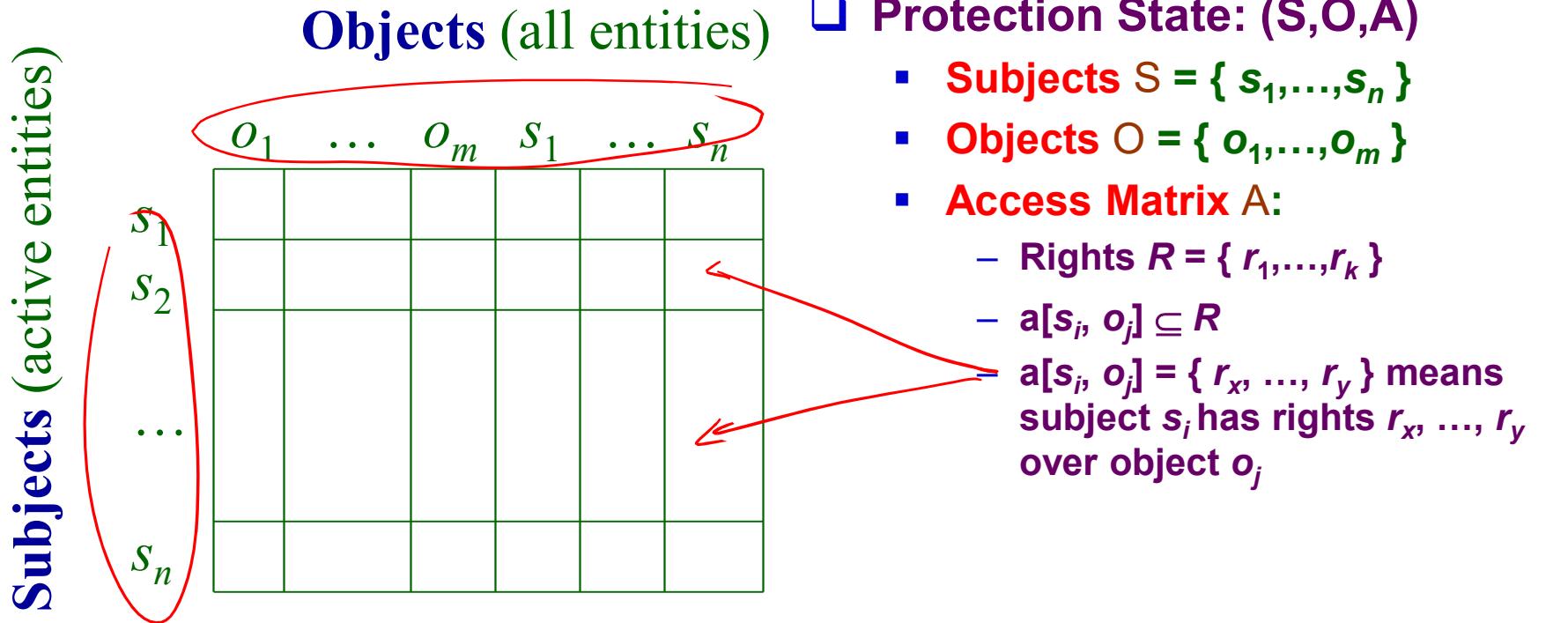
Example Security Models

- Access Control Matrix
 - Bell LaPadula
 - Biba Integrity
 - Chinese Wall
 - Clinical Information Systems Security (CISS) Policy
 - Clark-Wilson
-

Access Control Matrix (ACM) Model

- Characterizes access right relationships between subjects and objects
- Specification in the form of “Matrix” framework
 - A row shows the rights of one entity (over the other entities in the protection system)
 - A column shows the rights over one entity (by other entities in the protection system)

ACM Representation



Example ACM 1

□ Processes: p_1, p_2

↔ S & O

□ Files: f_1, f_2

↔ O

□ Rights: r, w, x, a, o

↔ a [s,o]

	f_1	f_2	p_1	p_2
p_1	rwo	r	$\square o$	w
p_2	a	ro	r	$\square o$

Example ACM 2

- Subjects & Objects: *telegraph, nob, toadflax*  **S & O**
- Rights: *own, ftp, nfs, mail*  **a [s,o]**

	<i>telegraph</i>	<i>nob</i>	<i>toadflax</i>
<i>telegraph</i>	own	ftp	ftp
<i>nob</i>		ftp, nfs, mail, own	ftp, nfs, mail
<i>toadflax</i>		ftp, mail	ftp, nfs, mail, own

- The subject *telegraph* is a personal computer with *ftp* client but no servers, so neither of the other systems can access it, but it can *ftp* to them. The subject *nob* is configured to provide NFS service to a set of clients that does not include the host *toadflax*, and both systems will exchange mail with any host and allow any host to use *ftp*.

Exercise 1

- Users:** Alice, Bob, Cyndy
- Files:** alicerc, bobrc, cyndyrc
- Rights:** read, write, own, execute

↔ ↔ S
↔ ↔ O
↔ ↔ a [s,o]

	alicer	bobrc	cyndyrc
Alice	read, write, own, execute	read	
Bob	read	read, write, own, execute	
Cyndy	read	read, write	read, write, own, execute

Exercise 1

- Users:** Alice, Bob, Cyndy
- Files:** alicerc, bobrc, cyndyrc
- Rights:** read, write, own, execute

↔ ↔ ↔ S
↔ ↔ ↔ O
↔ ↔ ↔ a [s,o]

	alicerc	bobrc	cyndyrc
Alice	read, write, own, execute	read	read
Bob		read, write, own, execute	
Cyndy	read	read, write	read, write, own, execute

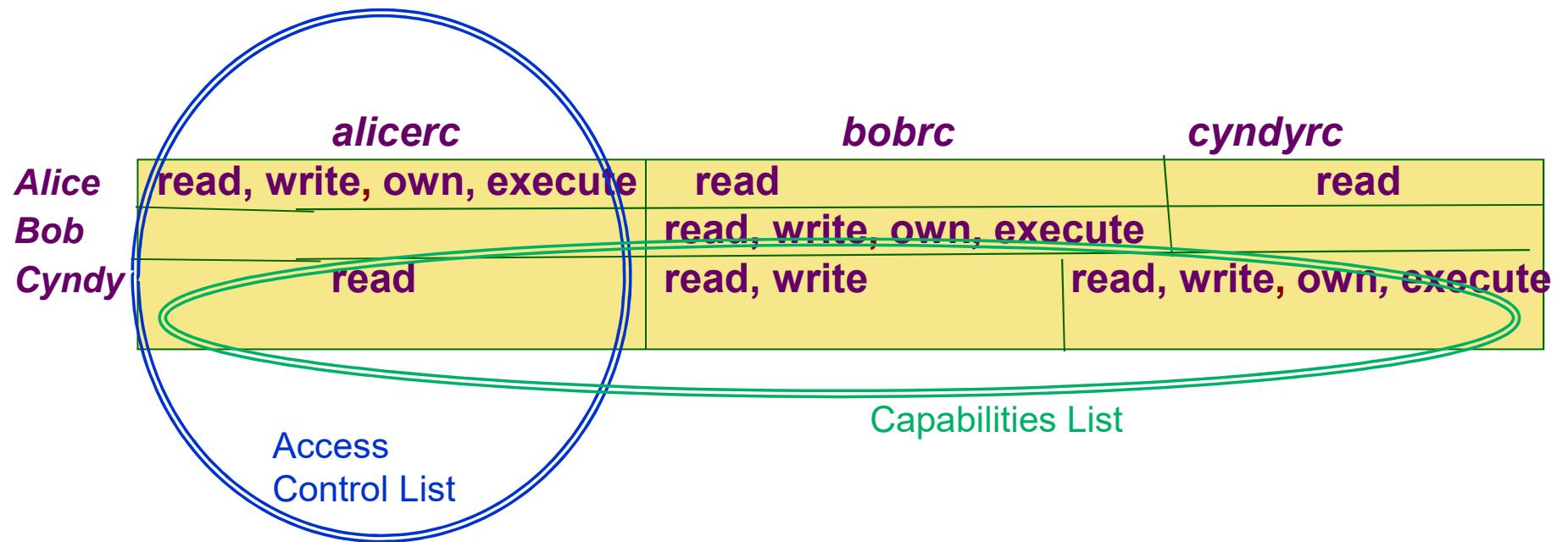
Reference Monitor and Validation Mechanism

- **Reference monitor (RM)** is access control concept of an abstract machine that mediates all accesses to objects by subjects.
- **Reference validation mechanism (RVM)** is an implementation of the reference monitor concept.
- **Properties**
 - RVM is Complete (always invoked)
 - RVM is Tamperproof (cannot be compromised)
 - RVM is Simple (verifiable)

~~ACM Implementation in Operating Systems~~

- In practical, since the access control matrix is “sparse”
 - implementation is either by row or by column
 - Row-wise implementation
 - Capabilities List *(per subject)*
 - Each subject has a list of objects it can access containing allowable ways to access
 - Column-wise implementation
 - Access Control List *(per object)*
 - Each object has a list of subjects that can access it containing allowable ways to be accessed

Example



~~Capabilities List~~

- A.k.a. Directory list
- Each subject is associated with a list of $(o, A[s, o])$ - which are assumed unforgeable
- Subject presents its “ticket” or “capability” to the Reference Monitor when making a request
 - ✓ Easier implementation
 - ✓ Easier to revoke access for certain subject
 - Diffuse concept of ownership (per object)
 - Difficult to revoke access for certain object

Access Control Lists

- Each object is associated with a list of $(s, A[s, o])$ - assumed unforgeable
- Reference Monitor must search all objects for each access request by a subject
 - ✓ Strong Ownership (per object)
 - ✓ Easier to revoke access for certain object
 - Difficult implementation
 - Difficult to revoke access for certain subject



Security & Safety

Safety

- Refers to abstract model
- Does a model that is safe w.r.t. all rights (privileges) guarantee a secure system?

Security

- Refers to implementation
- Does a secure (rather trusted) system corresponds to a model that is safe w.r.t. all rights?

Security Models –BLP, Biba

CSC 4575/5575
Information Assurance and Cryptography
Spring 2019

Sources:

Computer Security: Art and Science, Matt Bishop, Addison Wesley, 2003 [Chapters 5, 6]
Security in Computing, Pfleeger and Pfleeger, Prentice Hall, 2003

Bell-LaPadula Model (1)

- ❑ Cornerstone of many security models
 - ❑ Proposed by David E. Bell and Len LaPadula in 1973
 - To formalize DoD multilevel security policy
 - ❑ Focus on **secrecy** of information
 - Protects flow of information
 - ❑ Based on **hierarchical confidentiality**
 - Security levels arranged in linear ordering
 - Subjects have **security clearance $L(s)$**
 - Objects have **security classification $L(o)$**
 - ❑ Combines two (for reading and writing) **mandatory access control** rules (relationship of security levels) with **discretionary access control** (the required permission)
-

Reading Information

- Information flows UP
- “Reads up” disallowed, “reads down” allowed
 - Sometimes called “no reads up” rule
- Simple Security Condition (1)
 - Subject s can read object o iff $L(o) \leq L(s)$ and s has permission to read o
 - Subjects can *view content at or below their own security level*

No read



Information flows UP

Example

<i>security level</i>	<i>subject</i>	<i>object</i>
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Ulaley	Telephone Lists

Writing Information

- Information flows up**
- “Writes up” allowed, “writes down” disallowed
 - Sometimes called “no writes down” rule
- *-Property (1)
 - Subject s can write object o iff $L(s) \leq L(o)$ and s has permission to write o
 - Subjects can *create content at or above their own security level*

Information flows UP

No write



Example

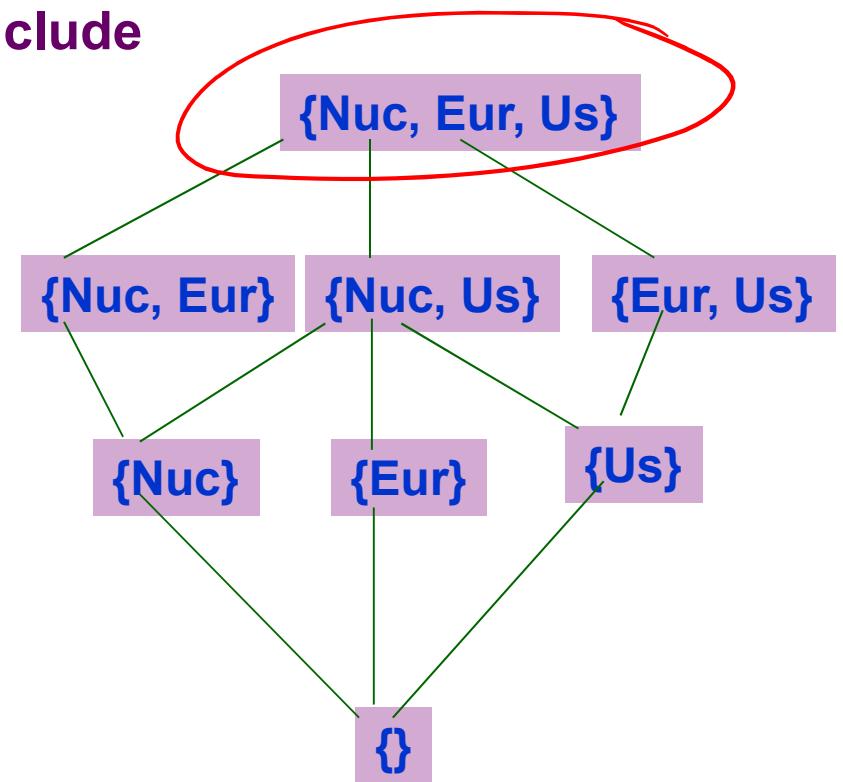
<i>security level</i>	<i>subject</i>	<i>object</i>
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Ulaley	Telephone Lists

Basic Security Theorem (1)

- If a system is initially in a secure state, and every transition of the system satisfies the simple security condition (1), and the *-property (1), then every state of the system is secure.
-

Bell-LaPadula Model (2)

- Expands notion of security level to include categories (types of information)
- Security level is
 - (*clearance/classification, category set*)
- Examples
 - (Top Secret, { NUC, EUR, US })
 - (Confidential, { EUR, US })
 - (Secret, { NUC, US })



Dominance relationship

- Captures both security level and category information
- $(L, C) \text{ dom } (L', C')$ iff $L' \leq L$ and $C' \subseteq C$
- Examples
 - (Top Secret, {NUC, US}) (Secret, {NUC})
 - (Secret, {NUC, EUR}) (Confidential, {US, EUR})
 - (Top Secret, {NUC}) (Confidential, {EUR})
 - (Secret, {EUR}) (Top Secret, {})

Reading Information

- “Reads up” disallowed, “reads down” allowed
- Simple Security Condition (2)
 - Subject s can read object o iff $L(s) \text{ dom } L(o)$ and s has discretionary read access to o

Writing Information

- “Writes up” allowed, “writes down” disallowed
- *-Property (2)
 - Subject s can write object o iff $L(o) \in \text{dom } L(s)$ and s has discretionary write access to o

Basic Security Theorem (2)

- If a system is initially in a secure state, and every transition of the system satisfies the simple security condition (2), and the *-property (2), then every state of the system is secure.
-

Problem

- Colonel has (Top Secret, {NUC, EUR}) clearance
- Major has (Secret, {EUR}) clearance
- Colonel cannot send message to Major
 - Violates no write down

Solution

- Define maximum, current security levels for (trusted) subjects
 - *maxlevel(s) dom curlevel(s)*
 - Colonel has *maxlevel* (Top Secret, { NUC, EUR })
 - Colonel sets *curlevel* to (Secret, { EUR })
- Example
 - Treat Major (Secret, {EUR} clearance) as an object (Colonel is writing to him/her)
 - Now *L(Major) dom curlevel(Colonel)*
 - Colonel can write to Major without violating “no writes down”

Biba Integrity Model

- Developed by Kenneth J. Biba in 1977
 - First attempt to model integrity constraints
 - Focus on preventing corruption of information
 - Based on multilevel integrity
 - The higher the integrity level, there is more confidence
 - that data is accurate and/or reliable
 - that a program will execute correctly
 - *Integrity levels are NOT confidentiality levels*
 - Dual of BLP model
 - Combines both MAC and DAC
-

Biba's Model

- **Information flows down**
 - **Simple integrity property**
 - $s \in S$ can read $o \in O$ iff $i(s) \leq i(o)$
 - No read down
 - Subjects can *view content at or above their own integrity level*
 - *** Integrity property**
 - $s \in S$ can write to $o \in O$ iff $i(o) \leq i(s)$
 - No write up
 - Subjects can *create content at or below their own integrity level*
-

~~Comparison of BLP and Biba~~

BLP –

- Focuses on confidentiality only
- Information flows up
- Applies to military/government sector mostly

Biba –

- Focuses on integrity only
- Information flows down
- Applies to commercial sector mostly

Both

- combines MAC and DAC
- intended for static policy
- do not address management of access control
- can be implemented at the same time

Security Models

CW, CISS, CLW

CSC 4575/5575
Information Assurance and Cryptography
Spring 2019

Sources:

Computer Security: Art and Science, Matt Bishop, Addison Wesley, 2003 [Chapter 7]

Chinese Wall Model

Problem:

- Tony advises American Bank about investments
- He is asked to advise Toyland Bank about investments
 - Conflict of interest

Brewer and Nash (1989)

Derived from British Law

Hybrid model – commercial policy

Characteristics

- **Conflict of interest**
- **Dynamic change of access rights**

Principle

- Combines commercial discretion with enforceable mandatory controls
- Data sanitized/unsanitized
 - Allows sanitized data to be viewed by everyone
 - For unsanitized data, access is controlled
- Dynamically establishes access rights of an user based on access history

Sanitization

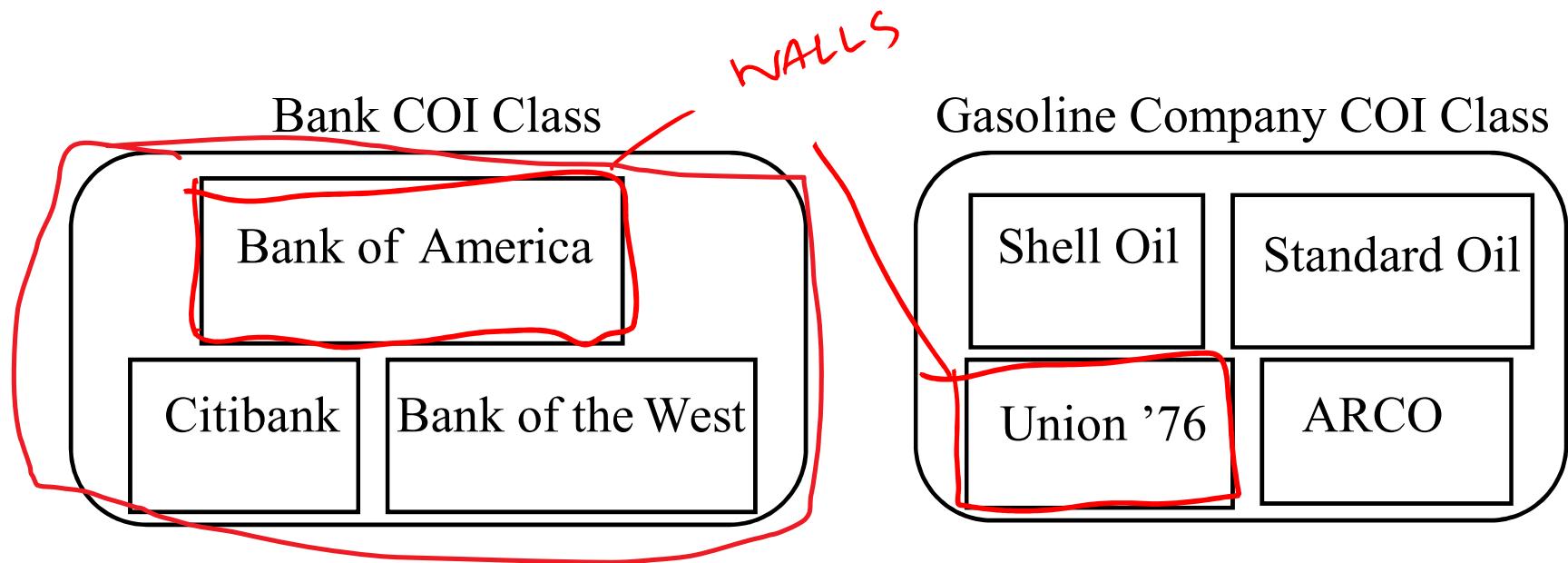
- Public information may belong to a CD
 - As is publicly available, no conflicts of interest arise
 - So, should not affect ability of analysts to read
 - Typically, all sensitive data removed from such information before it is released publicly (called *sanitization*)

Definitions

□ Three levels of abstraction

- *Objects*: items of information related to a company
- *Company dataset (CD)*: contains objects related to a single company
- *Conflict of interest class (COI)*: contains datasets of companies in competition
 - Assume: each object belongs to exactly one *COI* class

Example



Temporal Element

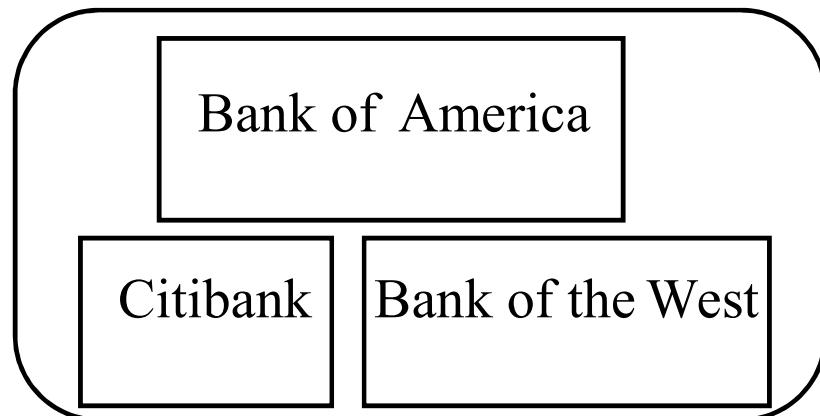
- If Anthony reads any CD in a COI, he can *never* read another CD in that COI
 - Possible that information learned earlier may allow him to make decisions later

CW-Simple Security Condition

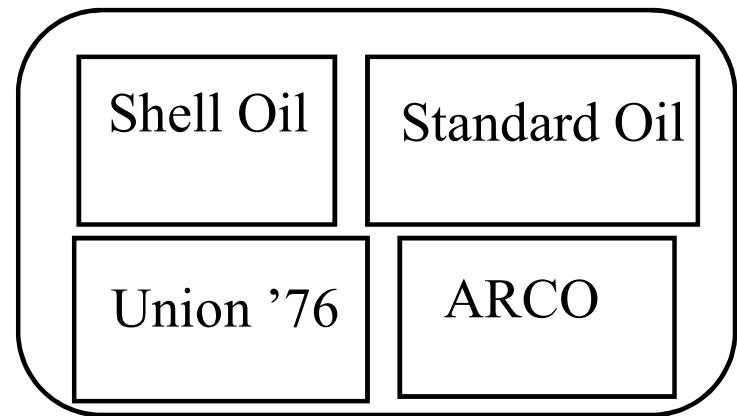
- Let $PR(S)$ be set of objects that S has already read
 - Initially, $PR(s) = \emptyset$, so initial read request granted
- s can read o iff either condition holds:
 1. There is an o' such that s has accessed o' and $CD(o') = CD(o)$
 - Meaning s has already read something in o' 's dataset
(ALREADY PERMITTED)
 2. For all $o' \in O, o' \in PR(s) \Rightarrow COI(o') \neq COI(o)$
 - Meaning anything s has read before does not belong to o 's conflict of interest class
(HAS NO KNOWLEDGE OF COMPETITORS)
 3. o is a sanitized object

Example

Bank COI Class



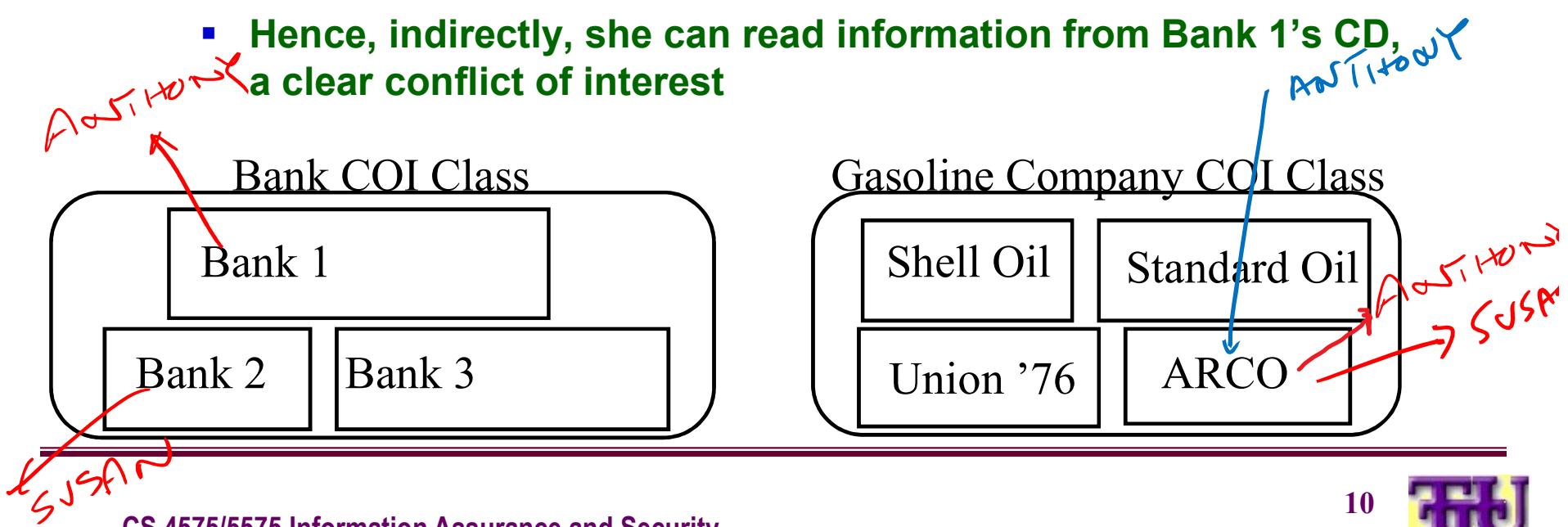
Gasoline Company COI Class



Writing

- Anthony, Susan work in same trading house
- Anthony can read Bank 1's CD, Arco Gas' CD
- Susan can read Bank 2's CD, Arco Gas' CD
- If Anthony could write to Arco Gas' CD, Susan can read it

- Hence, indirectly, she can read information from Bank 1's CD,
a clear conflict of interest

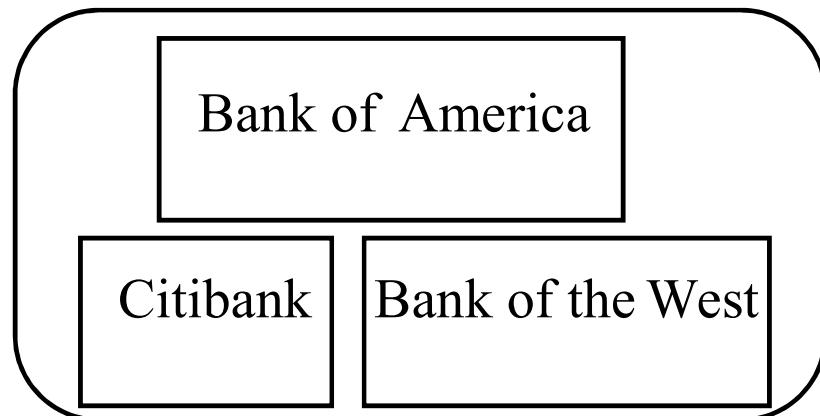


CW-*-Property

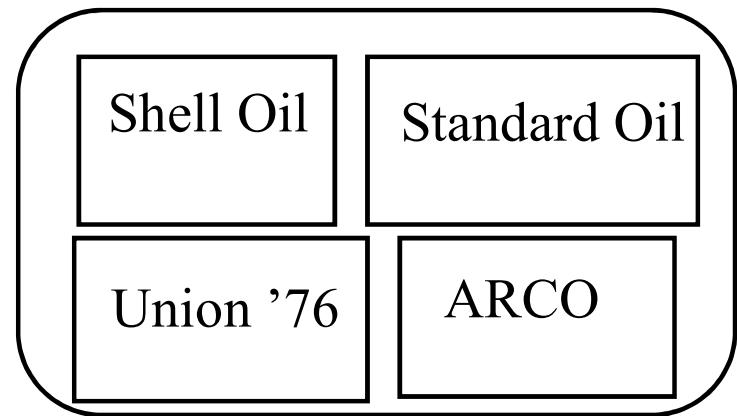
- **s can write to o iff both of the following hold:**
 1. The CW-simple security condition permits s to read o; and *(HAS READ PERMISSION)*
 2. For all *unsanitized* objects o' , if s can read o' , then $CD(o') = CD(o)$
(ONLY 'SECRET' KNOWLEDGE IS FROM THIS ONE)
- Says that s can write to an object if all the (unsanitized) objects it can read are in the same dataset

Example

Bank COI Class

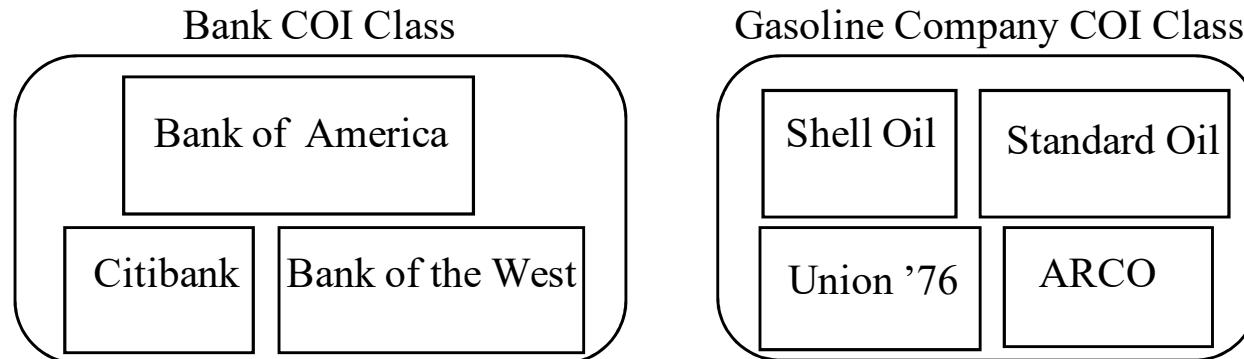


Gasoline Company COI Class



Implications

- Flow of sanitized information unrestricted
- Flow of unsanitized information confined to its own CD (within the wall)
- A subject is allowed access to only one CD in each COI
- In each COI, minimum number of subjects needed to access every object is equal to the number of CD. Any less than that will result in conflict of interest.
- Past history of access affects future accesses



Compare to Bell-LaPadula

- Fundamentally different
 - CW has no security labels, B-LP does
 - CW has notion of past accesses, B-LP does not
- Bell-LaPadula cannot track changes over time
 - Susan becomes ill, Anna needs to take over
 - C-W history lets Anna know if she can
- Freedom at first but access constraints change over time
 - Initially, subjects in C-W can read any object
 - Bell-LaPadula constrains set of objects that a subject can access
- Bell-LaPadula can give a subject full access/privilege
 - C-W cannot grant full access/privilege
 - Only one CD access from a COI

Criticism of Chinese Wall

- Some assumptions maybe unrealistic and/or flawed
 - Company datasets in different COIs might actually be in competition
 - Company datasets in the same COI might NOT actually be in competition
 - Therefore, define COI classes NOT based on business classes BUT based on common interests
-

Clinical Information Systems Security (CISS) Policy

- Anderson (1996)
- Prototypical HIPAA
- Hybrid model intended for medical records
 - Focus is Patient privacy and record integrity
- Entities:
 - Patient
 - subject of medical records (or agent)
 - Personal health information
 - data about patient's health or treatment enabling identification of patient
 - Clinician
 - health-care professional (or a group of professionals) with access to personal health information while in service

Assumptions and Principles

- Assumes health information involves 1 person at a time
 - Not always true
- Principles derived from medical ethics and practices
- Principles
 - Access
 - Audit
 - Creation
 - Deletion
 - Confinement
 - Aggregation
 - Enforcement

Access Principle 1 - ACL

- Each medical record has an access control list (ACL) naming the individuals or groups **who may read, update and append** information to the record. The system must restrict access to those identified on the access control list.
 - Idea is that clinicians need access, but no-one else. Auditors get access to copies, so they cannot alter records.

Access Principle 2 – Responsible Clinician

- One of the clinicians on the access control list must have the right to add other clinicians to the access control list.**
-

Access Principle 3 - Notification

- The responsible clinician must notify the patient of the names on the access control list whenever the patient's medical record is opened.
 - Except for situations given in statutes, or in cases of emergency, the responsible clinician must obtain the patient's consent (for treatment).
 - Patient must know of any violations of security.

Access Principle 4 - Audit

- The name of the clinician, the date, and the time of the access of a medical record must be recorded.
 - This is for auditing.
 - Don't delete information; update it.
- Similar information must be kept for deletions.

Creation Principle

→ CREATE

- A clinician may open a (new) record, with the clinician and the patient on the access control list. If a record is opened (created) as a result of a referral, the referring clinician may also be on the access control list.

Deletion Principle

- Clinical information **cannot be deleted from a medical record**
 - Unless patient died
 - Until appropriate time has passed.
 - Typically 8 years.

Confinement Principle

- Information from one medical record (A) may be appended to a different medical record (B) if and only if the access control list of the second record (B) is a subset of the access control list of the first (A).

- This keeps information from leaking to unauthorized users. All users have to be on the access control list.

$$A \rightarrow B \quad \text{iff} \quad B_{ACL} \subseteq A_{ACL}$$

Aggregation Principle

- Measures for preventing aggregation of patient data must be effective. In particular, a **patient must be notified if anyone is to be added to the access control list for the patient's record and if that person has access to a large number of medical records.**
 - Fear here is that a corrupt investigator may obtain access to a large number of records, correlate them, and discover private information about individuals which can then be used against patient's interest.

Enforcement Principle

- Any computer system that handles medical records must have a **subsystem that enforces the preceding principles**.
- The **effectiveness of this enforcement** must be subject to **evaluation by independent auditors**.
 - This policy has to be enforced, and the enforcement mechanisms must be auditable (and audited).

Compare to Bell-LaPadula

- CISS focuses on objects being accessed; B-LP on the subjects accessing the objects (capabilities)

Points to Ponder

Under CISS

- Who has ACCESS to records?
 - Under what circumstances?
- Who can CREATE record?
 - Under what circumstances?
- Who can APPEND TO record?
 - Under what circumstances?
- Who can DELETE record?
 - Under what circumstances?

Under CISS

- Under which circumstance, the patient needs notification?
 - What needs to be audited?
-

Clark-Wilson Model

- An integrity model better suited for commercial world
- Ensures the internal data is accurate and consistent to what it represents in the real world.
 - Internal consistency and external reality

Clark-Wilson Model

❑ Integrity

- Focused on users and their actions on data
 - Rather than data itself
- Defined by a set of constraints
 - Data in a *consistent* or valid state when it satisfies the constraints
 - Example: Bank
 - D today's deposits, W withdrawals, YB yesterday's balance, TB today's balance
 - Integrity constraint: $TB = D + YB - W$

Clark-Wilson Model

□ Characteristics

- Focus on: Data < Transactions > Users
 - Certain data items are constrained in that only certain processes may manipulate them.
 - Users are constrained in how they manipulate data.

Clark-Wilson Model Pillars

- Well formed transactions**
 - Transitions system from one valid state to another
 - Must preserve consistency

- Separation of duty**
 - Combinations of operations performed by different people
 - Certification and implementation/enforcement

Concepts

□ Data

- CDIs: Constrained Data Items
 - Data subject to integrity controls
- UDIs: Unconstrained Data Items
 - Data **not** subject to integrity controls

□ Procedures

- IVPs: Integrity Verification Procedures
 - Procedures that **test** the CDIs conform to the integrity constraints
- TPs: Transformation Procedures
 - Procedures that **take** the system from one valid state to another

Policy Rules for Integrity Assurance

- ❑ Policy defined by 5 certification and 4 enforcement rules
- ❑ Rules
 - Certification (security administration/system custodian)
 - Monitors integrity
 - Enforcement (system itself)
 - Preserves integrity

Certification Rules 1 and 2

CR1 When any IVP is run, it must ensure all CDIs are in a valid state

CR2 For some associated set of CDIs, a TP must transform those CDIs in a valid state into a (possibly different) valid state

- Defines *certified relation* that associates a set of CDIs with a particular TP

Enforcement Rules 1 and 2

ER1 The system must maintain the certified relations and must ensure that only TPs certified to run on a CDI manipulate that CDI.

- System must maintain, enforce certified relation

ER2 The system must associate a user with each TP and set of CDIs. The TP may access those CDIs on behalf of the associated user. The TP cannot access a CDI on behalf of a user who is not associated with that TP and CDI.

- System must enforce access based on user ID
(allowed relation)

Users and Rules

CR3 The allowed relations must meet the requirements imposed by the principle of separation of duty.

ER3 The system must authenticate each user attempting to execute a TP

- Type of authentication undefined, and depends on the instantiation
- Authentication *not* required before use of the system, but *is* required before manipulation of CDIs

Logging

CR4 All TPs must append enough information to reconstruct the operation to an append-only CDI.

- This CDI is the log

Handling Untrusted Input

CR5 Any TP that takes as input a UDI may perform only valid transformations, or no transformations, for all possible values of the UDI. The transformation either rejects the UDI or transforms it into a CDI.

Separation of Duty In Model

ER4 Only the certifier of a TP may change the list of entities associated with that TP. No certifier of a TP, or of an entity associated with that TP, may ever have execute permission with respect to that entity.

- Enforces separation of duty with respect to certified and allowed relations

A Flawed Implementation

- Rule ER2 sets up something similar to an access control list - a triple
`user_id, TPi, (CDIa, CDIb, ...)`

- An implementation of this policy used two relations/tables
[`user_id, TPi`] & [`TPi, (CDI's)`]

Example continued

□ Consider a system with

- 2 users (S1, S2)
- 2 TP's (TP1, TP2)
- 3 CDI (CDI1, CDI2, CDI3)

□ Policy is -

S1, TP1, CDI1

S1, TP2, CDI2

S2, TP1, CDI2

S2, TP2, CDI3

Policy
S1, TP1, CDI1
S1, TP2, CDI2
S2, TP1, CDI2
S2, TP2, CDI3

Resulting Bindings

User-Program bindings Program-Data bindings

(S1,TP1)

(TP1,CDI1)

(S1,TP2)

(TP1,CDI2)

(S2,TP1)

(TP2,CDI2)

(S2,TP2)

(TP2,CDI3)

Comparison to Biba

- Biba based on multilevel data integrity
- Clark-Wilson focuses on transaction integrity
- Biba
 - Subjects are given access to objects directly
 - No notion of certification rules; trusted subjects assume actions obey rules
 - Untrusted data made trusted by trusted subjects
- Clark-Wilson
 - Subjects are given access to objects through programs
 - Explicit certification requirements
 - Trusted entity certify *method* to upgrade untrusted data

Cryptography

CSC 4575/5575
Information Assurance and Cryptography
Spring 2019
Sources:

Introduction to Computer Security, Matt Bishop, Addison Wesley, 2003
Security in Computing, Pfleeger and Pfleeger, Prentice Hall, 2003
Cryptography & Network Security, B. Forouzan, McGraw Hill, 2007
CISSP Guide to Security Essentials, Chapter 5

Cryptography

- Art of secret writing**
- Concealing apparent meaning**
- Primarily deals with confidentiality**
 - Important integrity aspects
- Goals**
 - Should not be understandable to unauthorized party
 - Should prove correctness
 - Should prove origin
 - Should prove accountability

Early History

□ Non standard Hieroglyphs

- Egypt 2500 BC

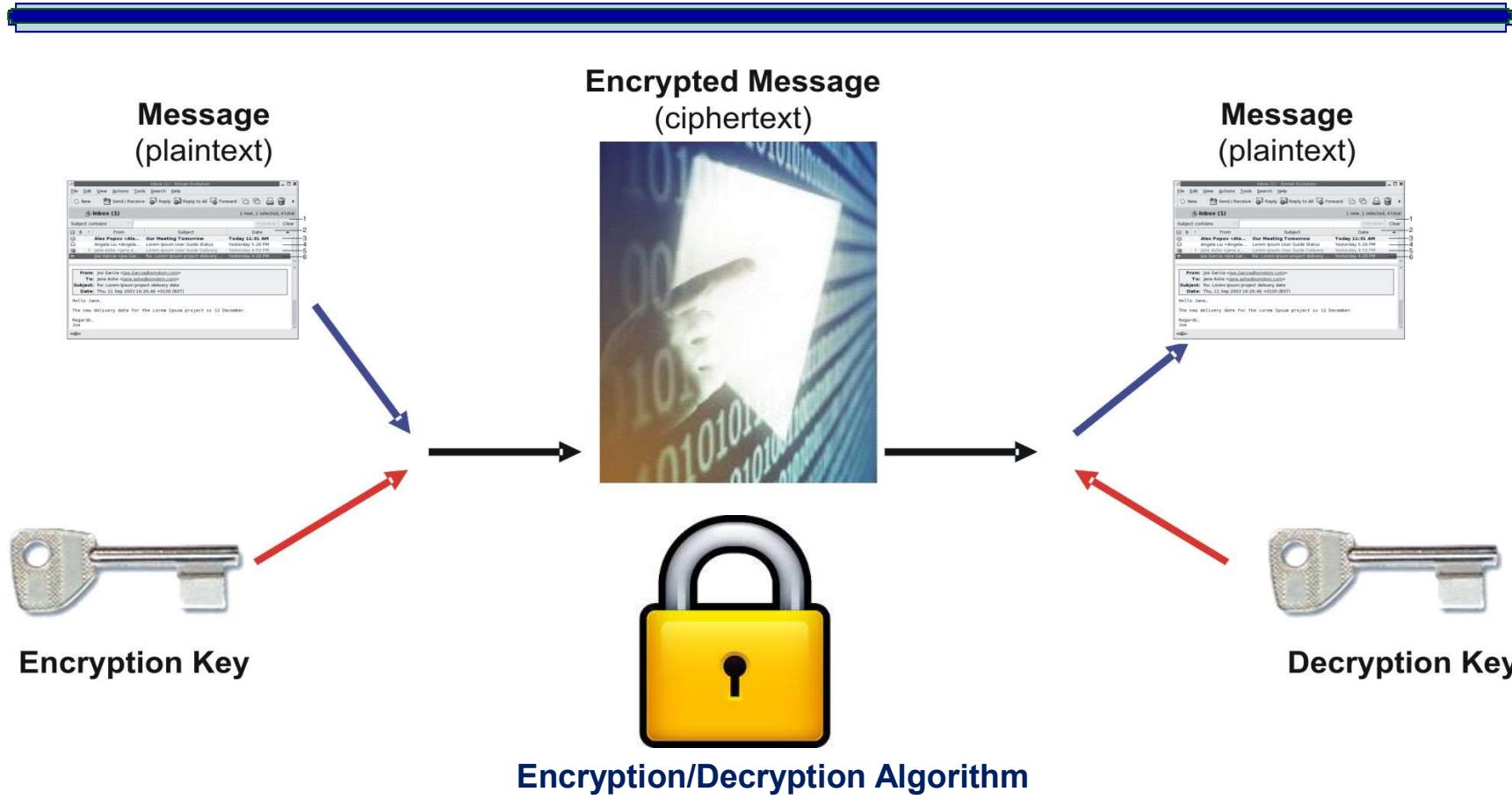
□ Atbash Cipher

- 500- 600 BC

□ Caeser Cipher (50-100 BC)

- Plaintext is HELLO WORLD
- Change each letter to another letter following a rule
 - Like X goes to A, Y to B, Z to C
 - and Ciphertext becomes KHOOR ZRUOG

Overview



http://swissquantum.idquantique.com/IMG/jpg/Graphic_crypto_principle.jpg

Terminology

- Plain text** Original text
 - Cipher text** Coded text
 - Crypto algorithm** Function to change plain text to cipher text
 - Key/cryptovariable** Used in coding/decoding text
 - Encryption** Collective term for encoding and enciphering
 - **Encoding** Coding entire words/phrases
 - **Enciphering** Coding each letter/symbol individually
 - Decoding/Deciphering/Decryption** Reverse
-

More Terminology

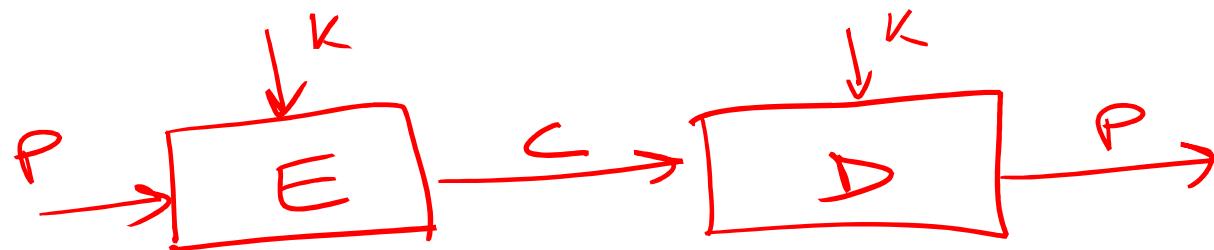
- **Cryptosystem** system for encryption and decryption
 - **Cryptography** use of encryption/decryption
 - Works for sender and receiver
 - **Cryptology** study of cryptosystem

 - **Cryptanalysis** attempts to break a cryptosystem
 - Works for interceptor
-

Cryptosystem

□ Quintuple $(\mathcal{E}, \mathcal{D}, \mathcal{P}, \mathcal{K}, C)$

- \mathcal{P} set of plaintexts
- \mathcal{C} set of ciphertexts
- \mathcal{E} set of encryption functions $e: \mathcal{P} \times \mathcal{K} \rightarrow \mathcal{C}$
- \mathcal{D} set of decryption functions $d: \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{P}$
- \mathcal{K} set of keys



Cryptanalysis

- Assume adversary knows algorithm used, but not key
- Four basic types of attacks:
 - *ciphertext only*
 - adversary has only ciphertext; goal is to find plaintext, possibly key
 - *known plaintext*
 - adversary has ciphertext with corresponding plaintext; goal is to find key
 - *chosen plaintext*
 - adversary may supply specific plaintexts of interest and obtain corresponding ciphertext; goal is to find key
 - *adversary has access to cryptosystem on sender's (encryption) side*
 - *chosen ciphertext*
 - adversary may supply specific ciphertexts of interest and obtain corresponding plaintext; goal is to find key
 - *adversary has access to cryptosystem on receiver's (decryption) side*

Basis for Attacks

- Brute force
- Mathematical attacks
 - Based on analysis of underlying mathematics
- Language attacks
 - Adhoc
 - Based on guess, no solid principle

Break This One!

w k l v p h v v d j h l v q r w w r r k d u g w r e u h d n

Basis for Attacks

□ Language attacks

- Adhoc
 - Based on guess, no solid principle
- Statistical analysis
 - Examine ciphertext, correlate properties with language.
 - Using models of the language (frequency and distribution of letters) make assumptions
 - pairs of letters (digrams), triplets of letters (trigrams), word boundaries, etc.

Types of Encryption/Decryption

□ Based on

- How keys are shared/distributed
- Key size
- How plain text is transformed into cipher text (operation)
- How plain text/cipher text is processed

Classical/Symmetric Cryptography

□ Sender, receiver share common key

- Keys may be the same
- Trivial to derive from one another

Based on Operation

- Two basic types

- Transposition ciphers
- Substitution ciphers
 - Combinations are called *product ciphers*

Transposition Cipher

- Rearrange letters in plaintext to produce ciphertext (letters themselves are NOT changed)
- Goal is diffusion
- Slower
- Example (Rail-Fence/Columnar Cipher)
 - Plaintext is HELLO WORLD
 - Rearrange as

HLOOL

ELWRD

- Ciphertext is HLOOL ELWRD

Attacking the Cipher

- Most common English Digrams (2-grams)
 - Most common English Trigrams
 - Assumption: If 1-gram frequencies match English frequencies, but other n -gram frequencies do not, probably transposition
 - ~~Anagramming~~ Technique for Cryptanalysis
 - Correlating observed pattern with known properties and rearranging letters to form n -grams with highest frequencies
-

Example

- Ciphertext: HLOOLELWRD
 - Get frequencies of English letters
 - Example: <http://www.cryptograms.org/letter-frequencies.php>
 - Find frequencies of 2-grams beginning with H
 - HE 0.0305
 - HO 0.0043
 - HL, HW, HR, HD < 0.0010
 - Frequencies of 2-grams ending in H
 - WH 0.0026
 - EH, LH, OH, RH, DH ≤ 0.0002
 - Implies E follows H
-

Example

- Ciphertext is HLOOL ELWRD
- Arrange so the H and E are adjacent

HE H L O O L
LL
OW OR E L W R D
OR
LD

- Read off across/, then down/, to get original plaintext Down Across

Substitution Ciphers

- Change/replace characters in plaintext to produce ciphertext
 - Goal is confusion
 - Faster
 - Example (Cæsar cipher)
-

Formal Representation of Caeser

□ Cæsar cipher for English Alphabets

- $\mathcal{P} = \{ \text{sequences of letters} \}$
- $\mathcal{K} = \{ i \mid i \text{ is an integer and } 0 \leq i \leq 25 \}$
- $\mathcal{E} = \{ E_k \mid k \in \mathcal{K} \text{ and for all letters } p,$

$$E_k(p) = (p + k)$$

- $\mathcal{D} = \{ D_k \mid k \in \mathcal{K} \text{ and for all letters } c,$

$$D_k(c) = (26 + c - k) \bmod 26 \}$$

To KEEP
WITHIN RANGE
 $0 \leq \text{result} \leq 25$

TO PREVENT
NEGATIVE
RESULT
IF $c < k$)

Attacking the Cipher

□ Exhaustive search

- If the key space is small enough, try all possible keys until you find the right one
- Cæsar cipher has 26 possible keys

□ Statistical analysis

- Compare to model of English

Observation

- COLLECT MOMENTS NOT THINGS
 - |
 - |
 - |
 - / / /
- JVSSLJA TVTLUAZ UVA AOPUNZ

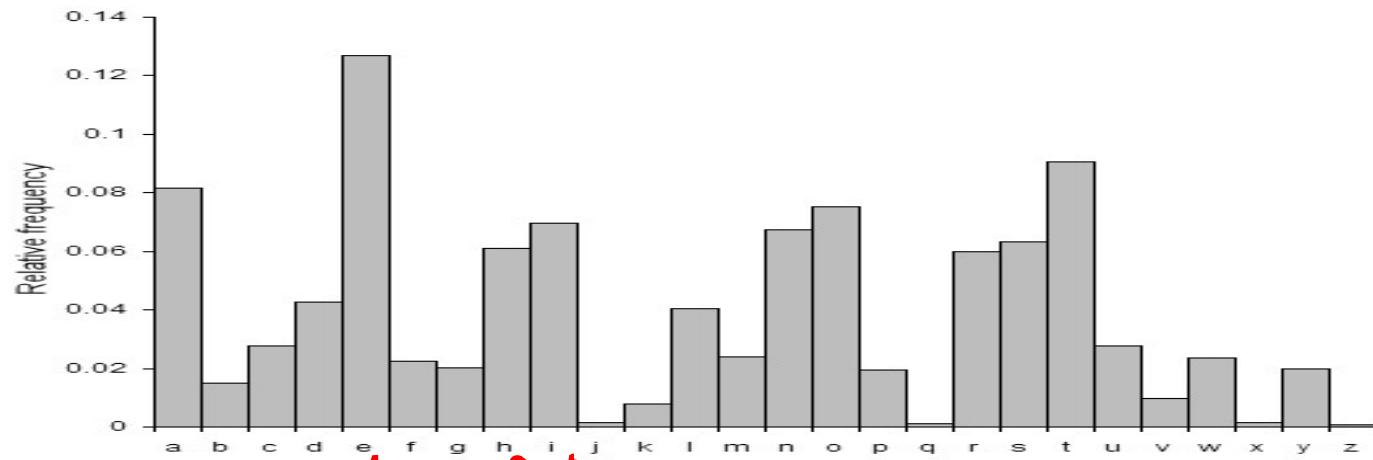
Statistical Attack

- Compute frequency of each letter in ciphertext
(KHOOR ZRUOG):

G	0.1	H	0.1	K	0.1	O	0.3
R	0.2	U	0.1	Z	0.1		

ETANOS

- Apply model of English language



<http://en.wikipedia.org/wiki/Image:English-slf.png>

f(x) Character Frequencies

(Chart XX)

a/0	0.080	h/7	0.060	n/13	0.070	t/19	0.090
b/1	0.015	i/8	0.065	o/14	0.080	u/20	0.030
c/2	0.030	j/9	0.005	p/15	0.020	v/21	0.010
d/3	0.040	k/10	0.005	q/16	0.002	w/22	0.015
e/4	0.130	l/11	0.035	r/17	0.065	x/23	0.005
f/5	0.020	m/12	0.030	s/18	0.060	y/24	0.020
g/6	0.015					z/25	0.002

Statistical Analysis

- $\varphi(i)$ correlation of frequency of letters in ciphertext with corresponding letters in English, assuming key is i
 - $\varphi(i) = \sum_{0 \leq c < 25} f(c)f'(e - i)$

Ciphertext is KHOOR ZRUOG

↑↑↑↑↑↑↑↑=X

Correlation: $\varphi(i)$ for $0 \leq i \leq 25$

i	$\varphi(i)$	i	$\varphi(i)$	i	$\varphi(i)$	i	$\varphi(i)$
0	0.0482	7	0.0442	13	0.0520	19	0.0315
1	0.0364	8	0.0202	14	0.0535	20	0.0302
2	0.0410	9	0.0267	15	0.0226	21	0.0517
3	0.0575	10	0.0635	16	0.0322	22	0.0380
4	0.0252	11	0.0262	17	0.0392	23	0.0370
5	0.0190	12	0.0325	18	0.0299	24	0.0316
6	0.0660					25	0.0430

The Result

- Most probable keys, based on ϕ :

- $i = 6, \phi(i) = 0.0660$
 - plaintext EBIIL TLOLA
- $i = 10, \phi(i) = 0.0635$
 - plaintext AXEEH PHKEW
- $i = 3, \phi(i) = 0.0575$
 - plaintext HELLO WORLD
- $i = 14, \phi(i) = 0.0535$
 - plaintext WTAAD LDGAS



- Only English phrase found is for $i = 3$

- That's the key (3 or 'D')

Cæsar's Problem

Key is too short (**Mono-alphabetic**)

- Can be found by exhaustive search
- Statistical frequencies easily derived/found
- Less confusion with 1-to-1 mapping

So make it longer (**Poly-alphabetic**)

- Multiple letters in key
- Idea is to smooth the statistical frequencies to make cryptanalysis harder
- More confusion with 1-to-many mapping

Vigènere Cipher

- Substitution like Cæsar cipher, but polyalphabetic key
- Use a word/phrase, repeatedly
- Example
 - Message THE BOY HAS THE BALL
 - Key VIG
 - Encipher using Cæsar cipher for each letter:

Key	VIGVIGVIGVIGVIGV
plain	THEBOYHASTHEBALL
cipher	OPKWWE CIYOPKWI RG

Useful Terms

❑ **period:** length of key

- In earlier example, period is 3

❑ **tableau:** table used to encipher and decipher

- Vigènere cipher has key letters on top, plaintext letters on the left

Relevant Parts of Vigènere Tableau

PLAIN TEXT

KEY

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

- Table shown has relevant rows, columns only
- Example encipherments:
 - key V, letter T: follow V column down to T row (giving "O")
 - Key I, letter H: follow I column down to H row (giving "P")

Key

Vigènere Tableau

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

Attacking the Cipher

□ Cryptanalysis Approach

- Find out period/key or n
- Break message into n different parts
 - each part being enciphered using the same key letter
- Solve each part individually
 - Each becomes a Caeser cipher

Key	VIGVIGVIGVIGVIGV
plain	THEBOYHASTHEBALL
cipher	OPKWWEICIYOPKWIRG

Finding the Period

❑ Kaskski's Observation:

- Repetitions in the ciphertext occur when characters of the key appear over the same characters in the plaintext
- Period is a factor of distance between such repetitions

❑ Example:

Key	VIGVIGVIGVIGVIGV
plain	THEBOYHASTHEBALL
cipher	OPKWWEICIYOPKWIRG

Note the key and plaintext line up over the repetitions. As distance between repetitions is 9, the period is a factor of 9 (that is, 1, 3, or 9)

The Target Cipher

- We want to break this cipher:

ADQYS MIUSB OXKKT MIBHK IZOOO
EQOOG IFBAG KAUMF VVTAA CIDTW
MOCIO EQOOG BMBFV ZGGWP CIEKQ
HSNEW VECNE DLAAV RWKXS VNSVP
HCEUT QOIOF MEGJS WTPCH AJMOC
HIUIX

Repetitions in Example

LETTERS	START	END	DISTANCE	FACTORS
MI	5	15	10	2, 5
OO	22	27	5	5
OEQOOG	24	54	30	2, 3, 5
FV	39	63	24	2, 2, 2, 3
AA	43	87	44	2, 2, 11
MOC	50	122	72	2, 2, 2, 3, 3
QO	56	105	49	7, 7
PC	69	117	48	2, 2, 2, 2, 3
NE	77	83	6	2, 3
SV	94	97	3	3
CH	118	124	6	2, 3

Estimate of Period

- OEQOOOG is probably not a coincidence
 - It's too long for that
- Most others (7/10) have 2 in their factors
- Almost as many (6/10) have 3 in their factors
- Begin with period of $2 \times 3 = 6$
 - (Greatest Common Divisor of two of the largest repetitions, OEQOOOG and MOC)

Validating the Period

- Index of coincidence (IC) is probability that two randomly chosen letters from ciphertext will be the same
 - Greater value of IC means greater probability, hence smaller key
- $$IC = [n(n - 1)]^{-1} \sum_{0 \leq i \leq 25} [F_i(F_i - 1)]$$
 - where n is length of ciphertext and F_i the number of times character i occurs in ciphertext

$$\text{Sum of } F_i = n$$

Calculate IC for KHOOR ZRUOG

Compute IC

- Tabulated results publicly available for IC in English for different periods:

1	0.066	3	0.047	5	0.044		
2	0.052	4	0.045	10	0.041	Large	0.038

- In example, IC for whole text = 0.043
 - Indicates a key of slightly more than 5

Splitting Into Alphabets

alphabet 1:

ICs

#1, 0.069;

alphabet 2:

#2, 0.078;

alphabet 3:

#3, 0.078;

alphabet 4:

#4, 0.056;

alphabet 5:

#5, 0.124;

alphabet 6:

#6, 0.043

ADQYS MIUSB OXKKT MIBHK IZOOO
EQOOG IFBAG KAUMF VVTAA CIDTW
MOCIO EQOOG BMBFV ZGGWP CIEKQ
HSNEW VECNE DLAAV RWKXS VNSVP
HCEUT QOIOF MEGJS WTPCH AJMOC
HIUIX

Indicate all alphabets have period 1, except #4 and #6; assume statistics off

Letter frequencies are (H high, M medium, L low):

HMMMHMMHHMMMHMLHHHMLLLLL

Frequency Examination

- alphabet 1:
- alphabet 2:
- alphabet 3:
- alphabet 4:
- alphabet 5:
- alphabet 6:

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
3	1	0	0	4	0	1	1	3	0	1	0	0	1	3	0	0	1	1	2	0	0	0	0	0	
1	0	0	2	2	2	1	0	0	1	3	0	1	0	0	0	0	0	1	0	4	0	4	0	0	
1	2	0	0	0	0	0	0	2	0	1	1	4	0	0	0	4	0	1	3	0	2	1	0	0	
2	1	1	0	2	2	0	1	0	0	0	1	0	4	3	1	0	0	0	0	0	0	2	1	1	
1	0	4	0	0	0	2	1	2	0	0	0	0	5	0	0	0	3	0	0	3	0	0	0	0	
0	1	1	1	0	0	2	2	3	1	1	0	1	2	1	0	0	0	0	0	3	0	1	0	1	

- alphabet 1: AIKHOIATTOBGEERNEOSAI**
 - alphabet 2: DUKKEFUAWEMGKWDWSUFWJU**
 - alphabet 3: QSTIQBMAMQBWQVLKVTMTMI**
 - alphabet 4: YBMZOAFCOOFPHEAXPQEPOX**
 - alphabet 5: SOIOOGVICOVCSVASHOGCC**
 - alphabet 6: MXBOGKVDIGZINNVCIJHH**
-

Letter frequencies are (H high, M medium, L low):

HMMMHMMHHMMMHMLHHM~~L~~LLL

Frequency Examination

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
3	1	0	0	4	0	1	1	3	0	1	0	0	1	3	0	0	1	1	2	0	0	0	0	0	0
1	0	0	2	2	2	1	0	0	1	3	0	1	0	0	0	0	0	0	1	0	4	0	4	0	0
1	2	0	0	0	0	0	0	2	0	1	1	4	0	0	0	4	0	1	3	0	2	1	0	0	0
2	1	1	0	2	2	0	1	0	0	0	0	1	0	4	3	1	0	0	0	0	0	0	2	1	1
1	0	4	0	0	0	2	1	2	0	0	0	0	5	0	0	0	3	0	0	3	0	0	0	0	0
0	1	1	1	0	0	2	2	3	1	1	0	1	2	1	0	0	0	0	0	3	0	1	0	1	0

→ A

→ I

→ V

alphabet 1: AIKHOIATTOBGEERNEOSAI

alphabet 2: DUKKEFUAWEMGKWDWSUFWJU

alphabet 3: QSTIQBMAMQBWQVLKVTMTMI

alphabet 4: YBMZOAFCOOFPHEAXPQEPOX

alphabet 5: SOIOOGVICOVCSVASHOGCC

alphabet 6: MXBOGKVDIGZINNVVICIJHH

Begin Decryption

- First matches characteristics of unshifted alphabet *KEY ALPH AND = A*
- Third matches if I shifted to A *3RD KEY ALPH = I = A*
- Sixth matches if V shifted to A *6TH V = V*
- Substituting plaintexts for ciphertexts for 1st, 3rd and 6th alphabets (red are substitutions)

ADIYS RIUKB OCKKL MIGHK AZOTO EIOOL IFTAG PAUEF
VATAS CIITW EOCNO EIOOL BMTFV EGGOP CNEKI HSSEW
NECSE DDAAA RWCXS ANSN HHEUL QONOF EEGOS WLPCM
AJEOC MIUAX

Determining 2nd Alphabet in Key

ADIYS RIUKB OCKKL MIGHK AZOTO EIOOL IFTAG
PAUEF VATAS CIITW EOCNO EIOOL BMTFV EGGOP CNEKI
HSSEW NECSE DDAAA RWCXS ANSN HHEUL QONOF EEGOS
WLPCM AJEOC MIUAX

3

- AJE in last line suggests “are”, meaning second alphabet key is S

2nd KEY ALPH = S

Determining 4th Alphabet in Key

ALIYS RICKB OCKSL MI GHS AZOTO
MIOOL INTAG PACEF VATIS CI ITE
EOCNO MIOOL BUTFV EGOOP CNESI
HSSEE NECSE LDAAA RECXS ANANP
HHECL QONON EEGOS ELPCM AREOC
MICAX

- MICAX in last line suggests “mical” (a common ending for an adjective), meaning fourth alphabet key is M;

4TH Key Alpha = M

Determining 5th Alphabet in Key (One way)

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
3	1	0	0	4	0	1	1	3	0	1	0	0	1	3	0	0	1	1	2	0	0	0	0	0	0
1	0	0	2	2	2	1	0	0	1	3	0	1	0	0	0	0	0	1	0	4	0	4	0	0	0
1	2	0	0	0	0	0	0	2	0	1	1	4	0	0	0	4	0	1	3	0	2	1	0	0	0
2	1	1	0	2	2	0	1	0	0	0	0	1	0	4	3	1	0	0	0	0	0	2	1	1	1
1	0	5	0	0	0	2	1	2	0	0	0	0	5	0	0	0	3	0	0	2	0	0	0	0	0
0	1	1	1	0	0	2	2	3	1	1	0	1	2	1	0	0	0	0	0	3	0	1	0	1	1

Letter frequencies are (H high, M medium, L low):

HMMMHMMHHMMMHMLHHHMLLL

Determining 5th Alphabet in Key (Another way)

ALIMS RICKP OCKSL AIGHS ANOTO MICOL INTOG PACET
VATIS QIITE ECCNO MICOL BUTTV EGOOD CNESI VSSEE
NSCSE LDOAA RECLS ANAND HHECL EONON ESGOS ELDCM
ARECC MICAL

- QI means that U maps into I, as Q is always followed by U, meaning fifth alphabet key is O;

The Key and Ciphertext after Splitting Into Alphabets in Key

- A alphabet 1: AIKHOIATTOBGEERNEOSAI
- S alphabet 2: DUKKEFUAWEMGKWDWSUFWJU
- T alphabet 3: QSTIQBMAMQBWQVLKVTMTMI
- M alphabet 4: YBMZOAFCOOFPHEAXPQEPOX
- O alphabet 5: SOIOOGVICOVCSVASHOGCC
- ✓ alphabet 6: MXBOGKVDIGZINNVCIJHH

The PlainText!

ALIME RICKP ACKSL AUGHS ANATO MICAL INTOS PACET
HATIS QUITE ECONO MICAL BUTTH EGOOD ONESI VESEE
NSOSE LDOMA RECLE ANAND THECL EANON ESSOS ELDOM
ARECO MICAL

Key: ASIMOV

One-Time Pad

- A Vigenère cipher with
 - random key
 - at least as long as the message
 - used only once
- Provides perfect secrecy IF the key is
 - Truly random
 - Never reused
 - Kept secret
- However
 - Not so much used because of key distribution problems

~~Good Cipher Characteristics~~

Shanon 1949

- Need for protection should determine cost/effort
- None or minimal complexity for usage
- Simple (reasonable) implementation
- Errors should not propagate
- Cipher text should not be longer than the original text

More ..

- Confusion such that by changing P attacker cannot predict whole C
- Diffusion such that any changes in P are spread out to much of C
- Based on sound principles
- Peer/expert reviewed
- Withstand “test of time”

DES

- Data Encryption Standard (FIPS 46-3)
- Most important classical cryptosystem
- History
 - 1973/74 call for proposal by NBS for a public encryption algorithm that could protect unclassified but sensitive information
 - High level of security/ easy to understand/ publishable/ available/adaptable/economical/efficient
 - Data Encryption Algorithm (DEA) was designed by IBM, based on Horst Feistel's Lucifer cipher, reviewed by NSA
 - Adopted by NBS/NIST in 1976
 - International standard for use in public and private sector

Overview of the DES

Symmetric cryptography algorithm

- Same algorithm for encryption/decryption
- Same key for encryption/decryption

A block cipher

- bit-oriented
- encrypts blocks of 64 bits using a 64 bit key
- outputs 64 bits of ciphertext



DES Input

1. Get Text = "Hello World!"

2. Convert to Binary

H = 01001000	W = 01010111
e = 01100101	o = 01101111
l = 01101100	r = 01110010
l = 01101100	l = 01101100
o = 01101111	d = 01100100
= 00100000	! = 00100001

3. Break into 64-bit blocks

block 1:	block 2:
0 1 0 0 1 0 0 0	0 1 1 1 0 0 1 0
0 1 1 0 0 1 0 1	0 1 1 0 1 1 0 0
0 1 1 0 1 1 0 0	0 1 1 0 0 1 0 0
0 1 1 0 1 1 0 0	0 0 1 0 0 0 0 1
0 1 1 0 1 1 1 1	padding
0 0 1 0 0 0 0 0	padding
0 1 0 1 0 1 1 1	padding
0 1 1 0 1 1 1 1	padding

<http://kathryneugent.com/des.html>

~~Stream and Block Cipher~~

□ Stream cipher

- Enciphers character by character
- Faster decoding
- Computational power can be issue for large plain text
- Low diffusion
- Low error propagation
- Susceptible to malicious insertion
- Example:
 - Caeser cipher

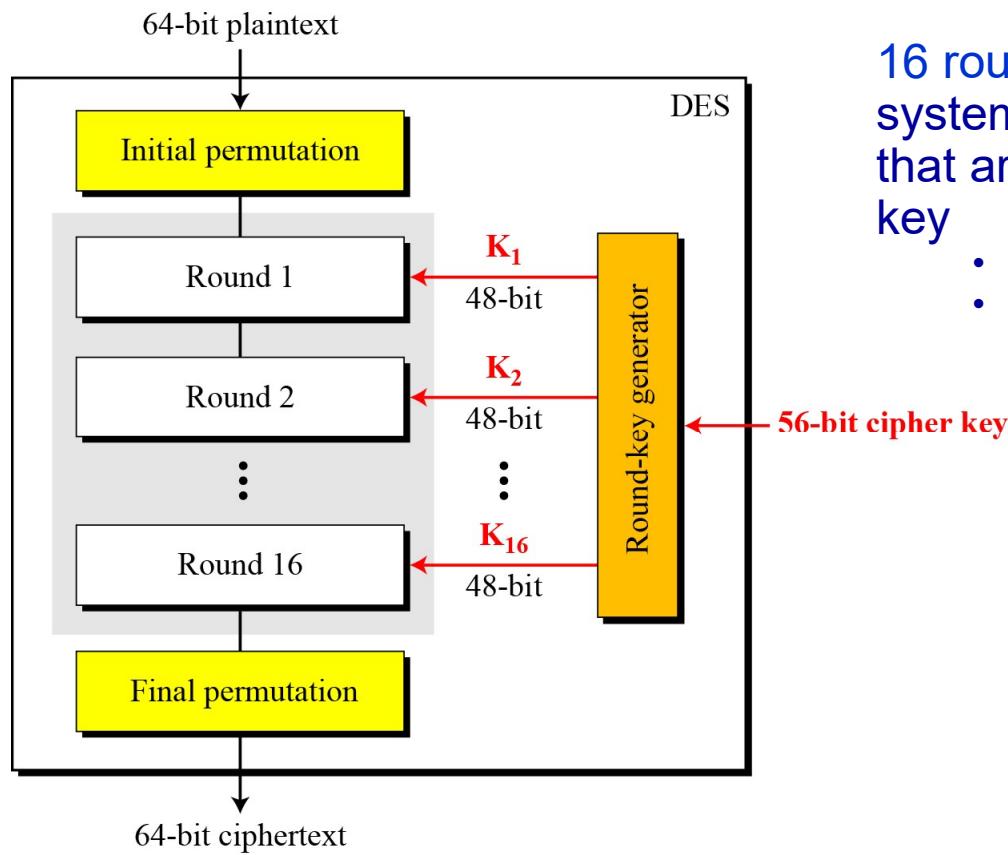
□ Block cipher

- Enciphers a group of characters each tin
- Slower decoding
- Computational power is less of an issue
- High diffusion
- High error propagation
- Immune to malicious insertions
- Example:
 - Rail Fence cipher, DES

Strength of the DES

- A product cipher
 - performs both substitution and transposition (a.k.a., permutation) on the bits
 - Employs both confusion and diffusion
- Feasible implementation
- Strong cryptographic properties (avalanche and completeness effect)

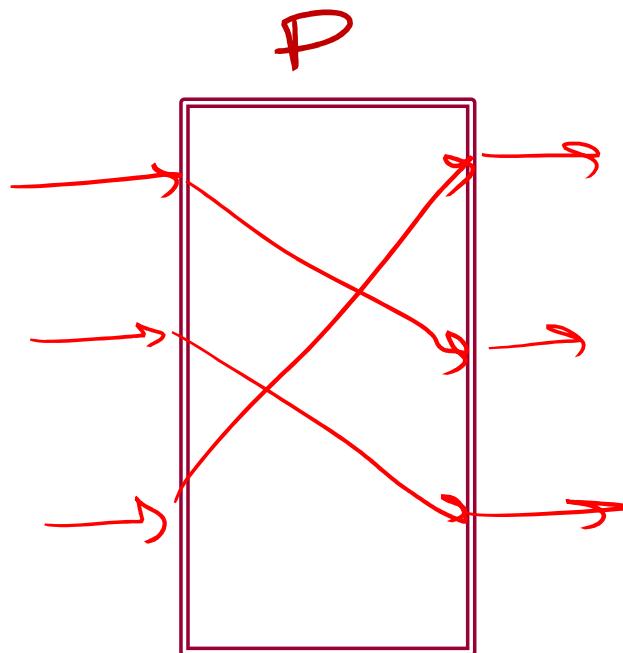
General Structure of DES



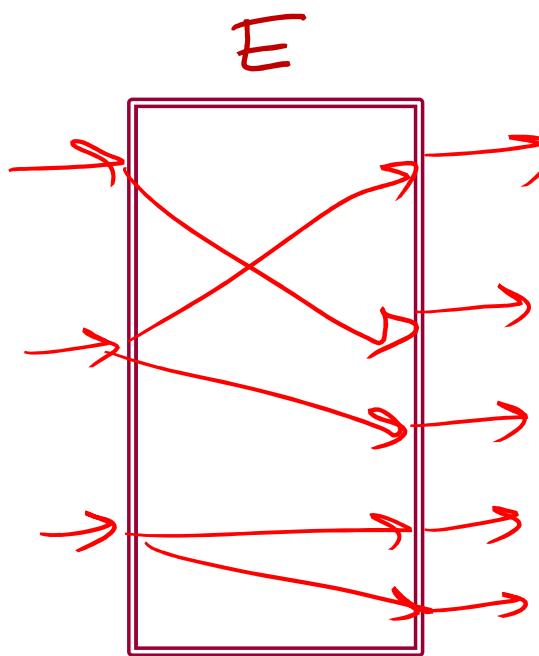
16 rounds (iterations) with 16 different system generated round keys (subkeys) that are derived from a single user supplied key

- Sequential rounds
- Subkeys used in reverse order for decryption

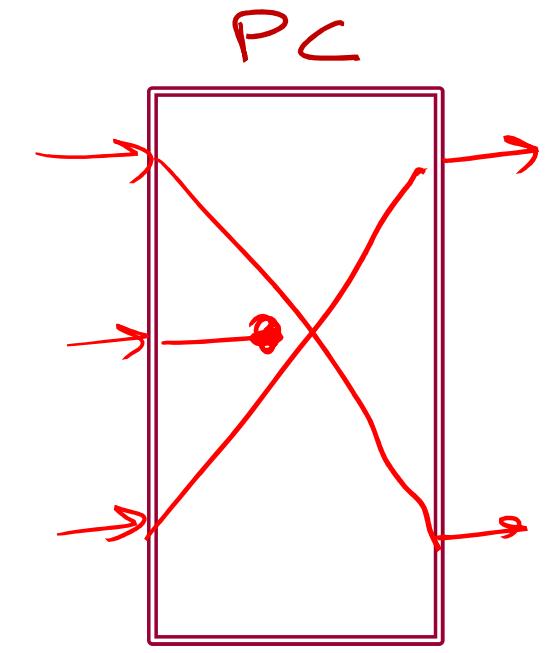
Types of Permutation



PERMUTATION
STRAIGHT

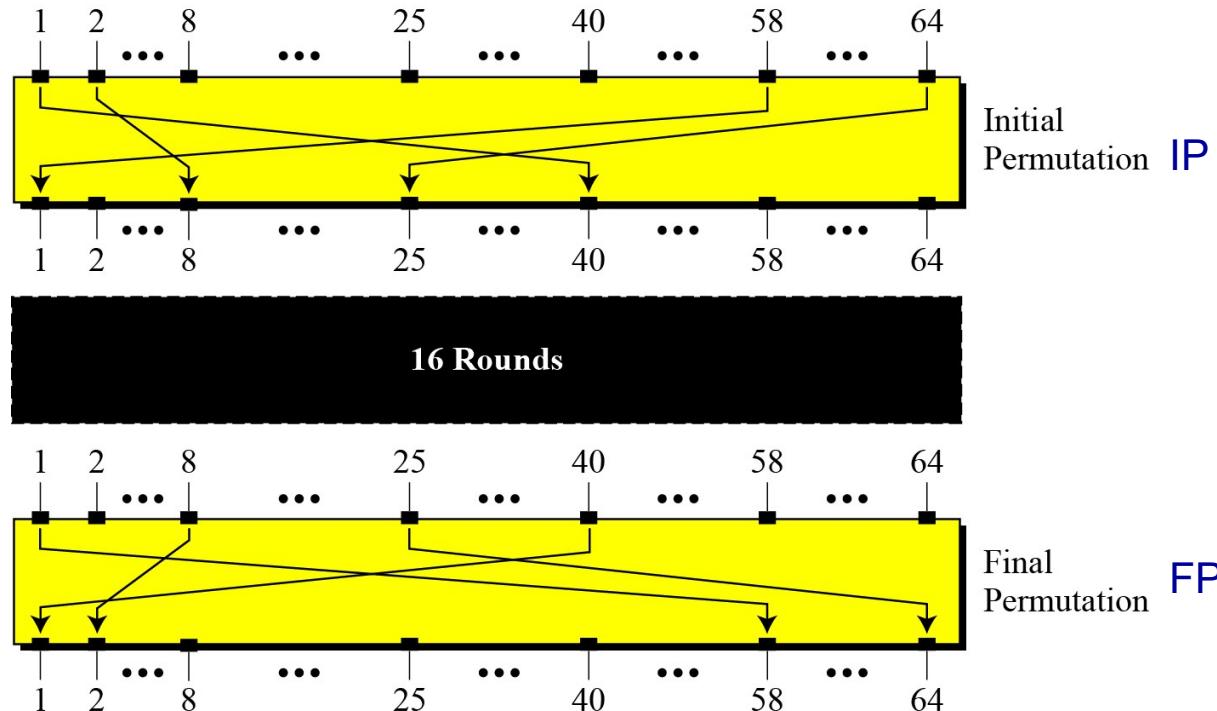


EXPANSION



PERMUTED
CHOICE
(COMPRESSION)

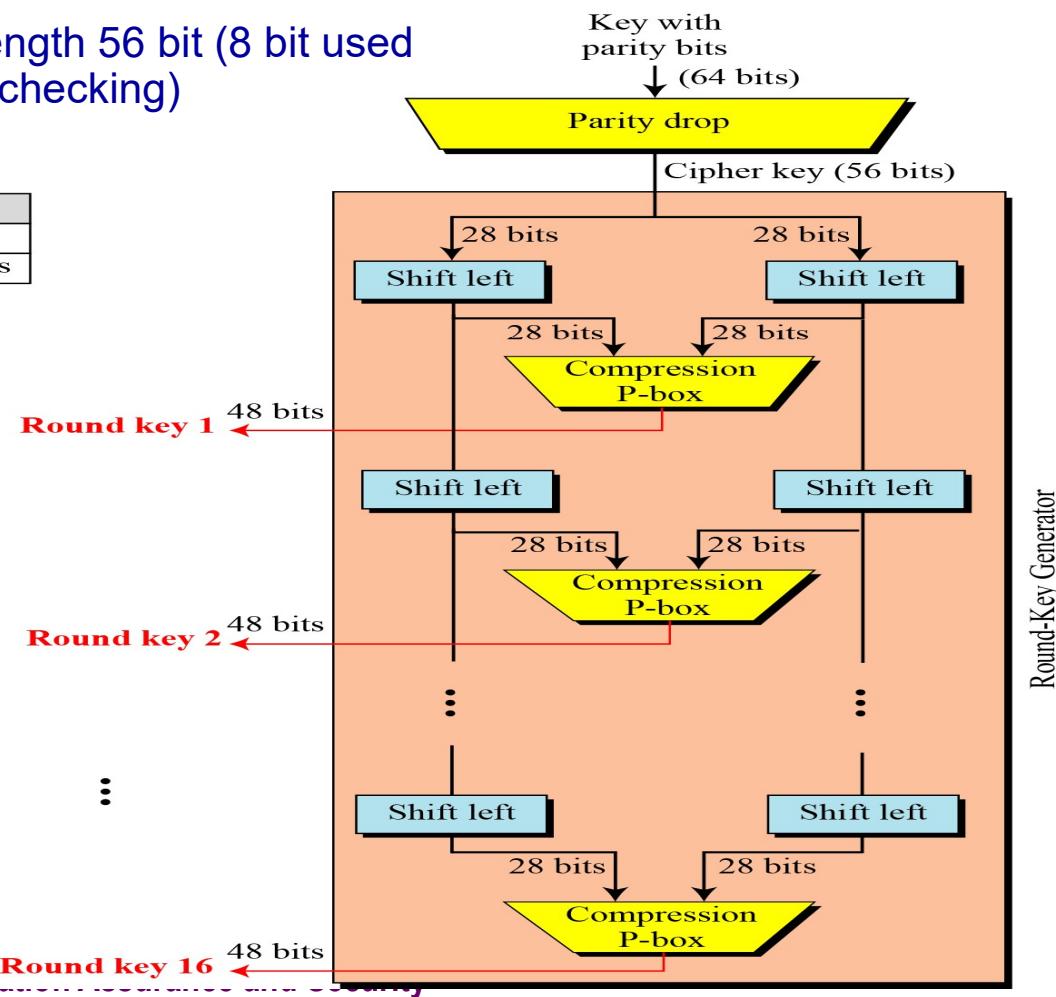
Initial and Final Permutation



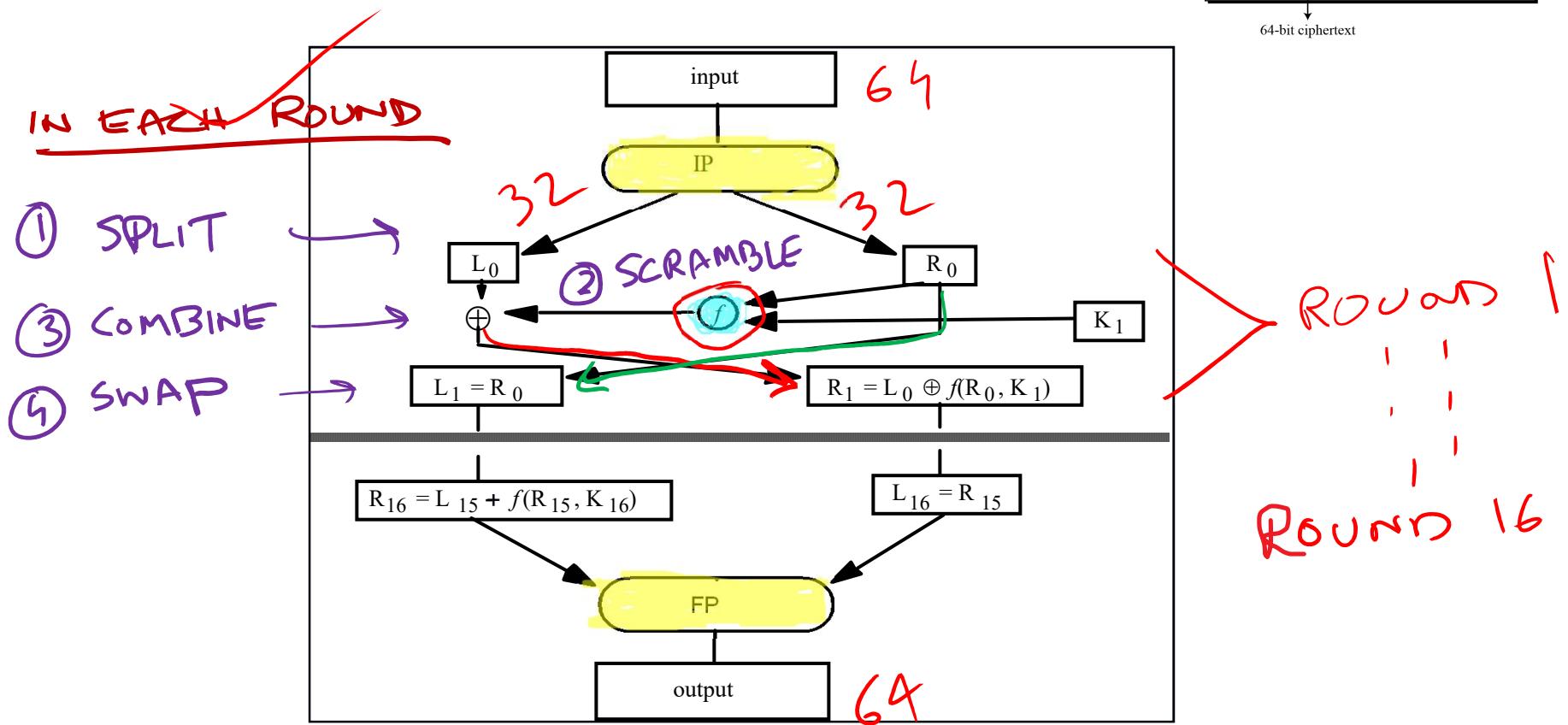
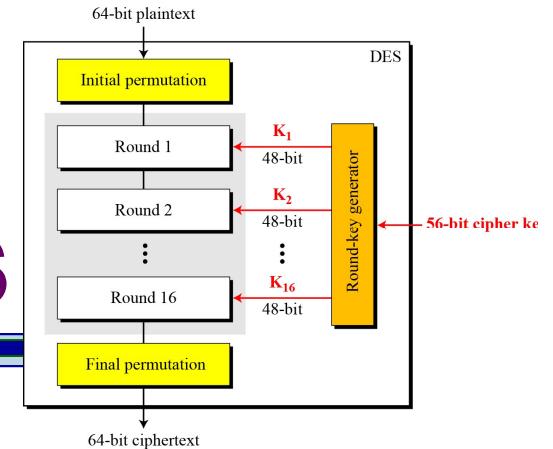
Key Generation in DES

Effective key length 56 bit (8 bit used for parity/error checking)

Shifting	
Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

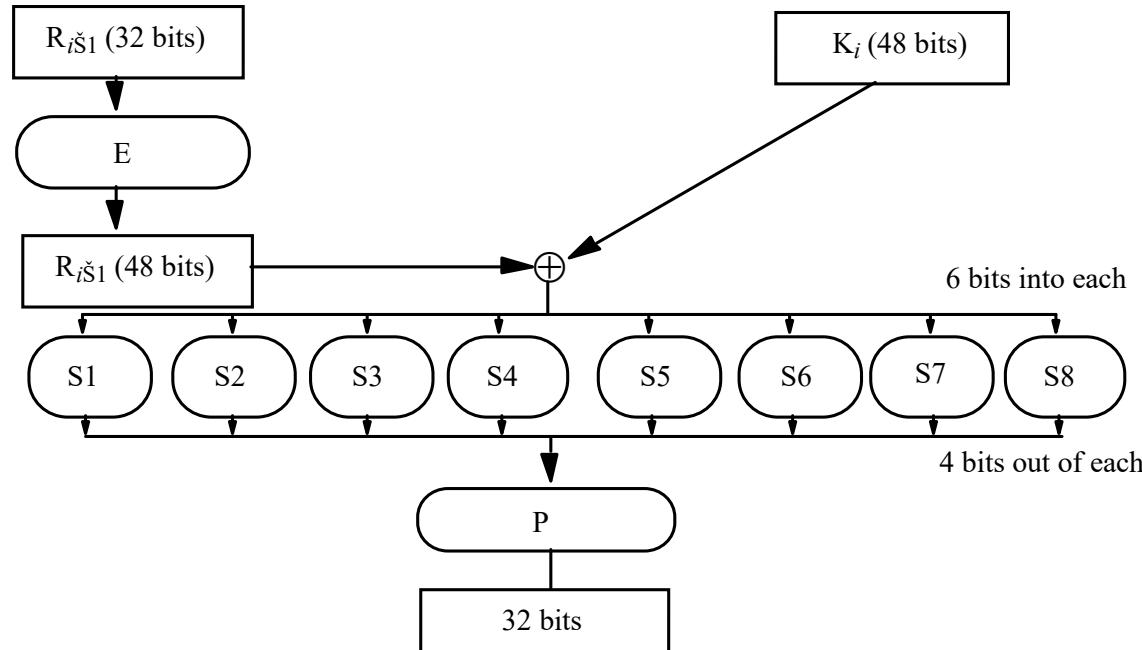


Workings of DES



Core of DES

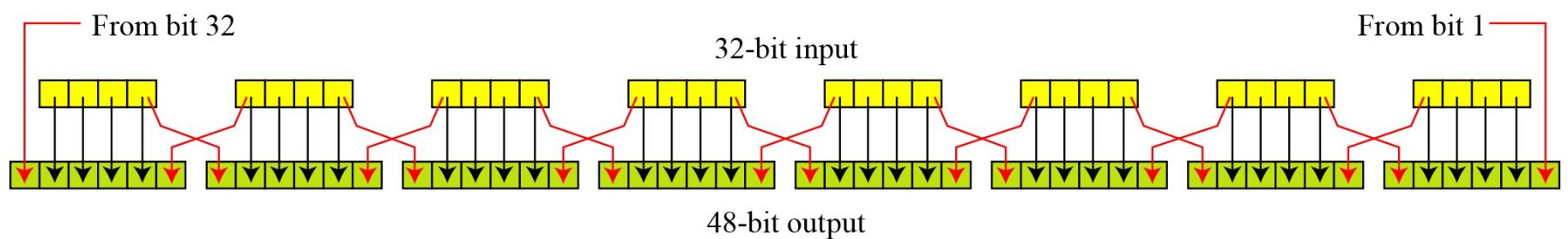
Feistel (f) Function



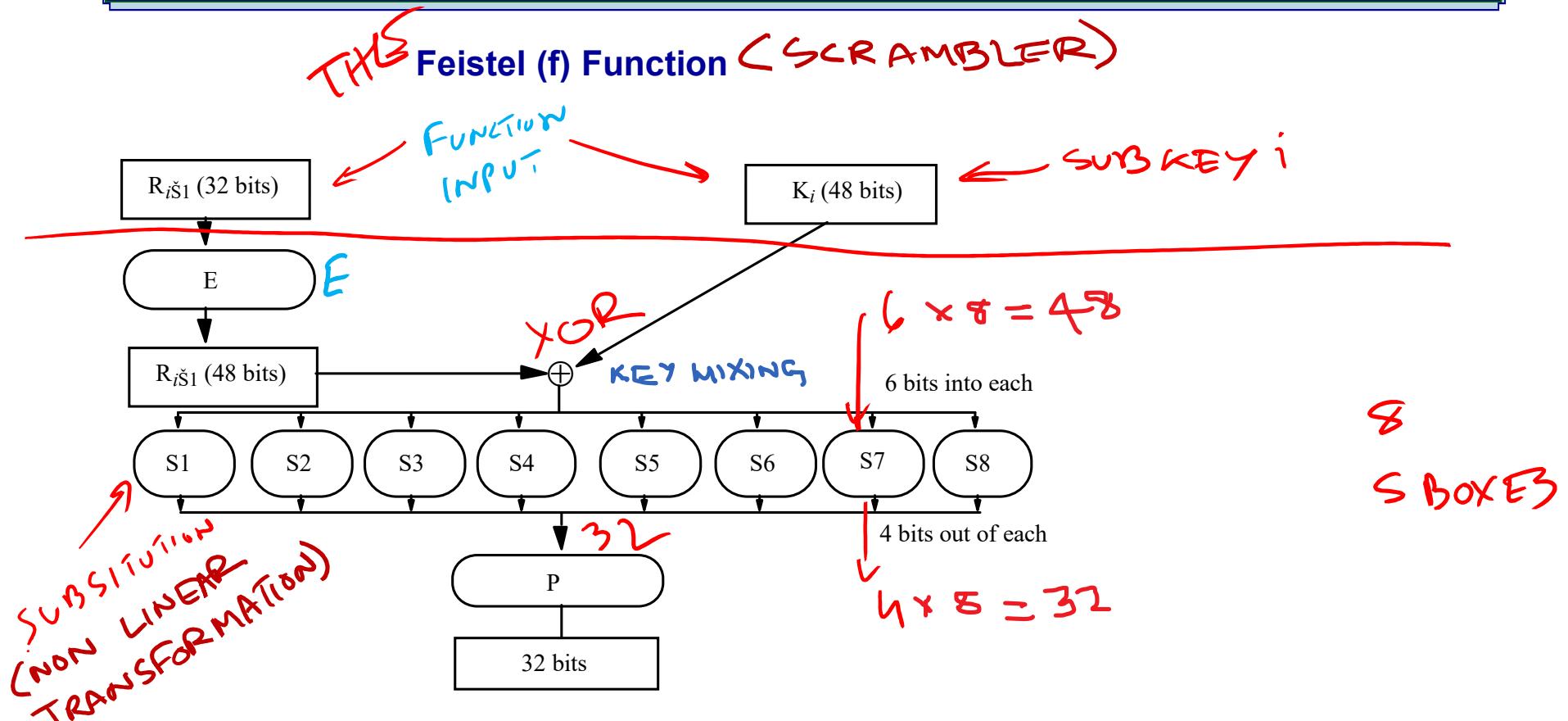
http://en.wikipedia.org/wiki/Data_Encryption_Standard

CS 4575/5575 Information Assurance and Security

DES Expansion Permutation (E)



Core of DES



http://en.wikipedia.org/wiki/Data_Encryption_Standard

CS 4575/5575 Information Assurance and Security

DES S-Box

Look up Table Example

□ For 5th S-Box

S ₅		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

□ If input is: 011011 output will be 1001

http://en.wikipedia.org/wiki/Substitution_box

Strong Cryptographic Properties of DES

□ Avalanche effect

- Small change in plain text or key makes significant change in the ciphertext.
- Example:
 - Plain text: 0000000000000000
 - Ciphertext: 7J388FNSWKR51234
 - Plain text: 0000000000000001
 - Ciphertext: 6DBWK3479SH00FJK2

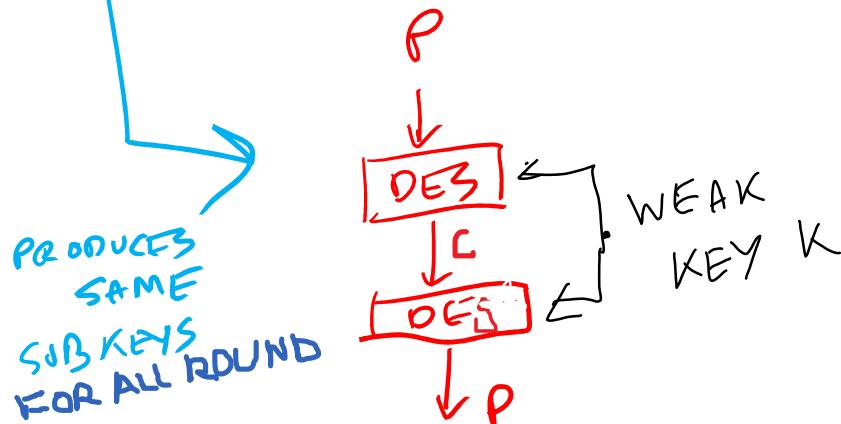
□ Completeness effect

- Each bit of the ciphertext depends on many bits of plaintext.

Weak Properties of DES

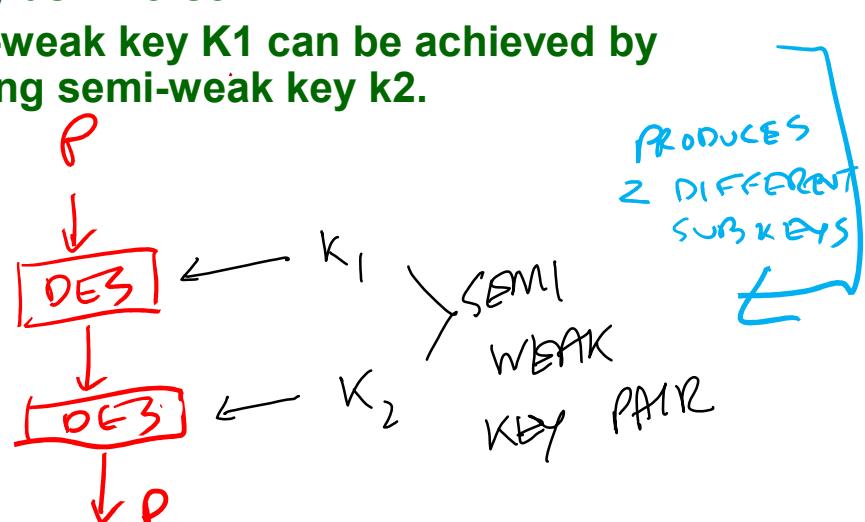
Out of 2^{56} keys

- 4 weak keys (all 1s, all 0s, half 1s and 0s, half 0s and 1s)
 - They are their own inverses $C = E(P, K)$ $P = E(C, K)$
 - Effect of decryption with a weak key can be achieved by encryption with the same .



Weak Properties of DES cont.

- 12 semi-weak keys (6 K_1, K_2 pairs) $C = E(P, K_1)$ $P = E(C, K_2)$
 - Each has another semi-weak key as inverse
 - Effect of decryption with a semi-weak key K_1 can be achieved by encryption with the corresponding semi-weak key K_2 .



Weak Properties of DES cont.

Complementation property

- Knowing half of DES keys, you can guess the other half
- Brute force attack effort could be reduced by a factor of 2^{55}

$$C = E(P, K) \quad \bar{C} = E(\bar{P}, \bar{K})$$

S-boxes exhibit irregular properties

- Did not randomize input properly
- Output of fourth S-box depends on input to third S-box

P-box (permutation) mystery

- No cryptographic significance of initial and final permutation

Controversy with DES

❑ Considered too weak

- Short key size (56 bit)
- Design decisions not public (initially)
 - “Suspicious” S-boxes
 - May have trapdoors

❑ Diffie, Hellman (1977) designed a parallel machine to break DES by brute force and predicted that in a few years technology would allow DES to be broken in days

- And it did
 - 1997, 3500 machines in parallel working for 4 months
 - 1998, DES Cracker in a few days
 - 1999, DES Cracker with distributed.net in less than a day

ESTIMATED COST
THEORETICAL
20 MILLION

Cryptanalysis for DES

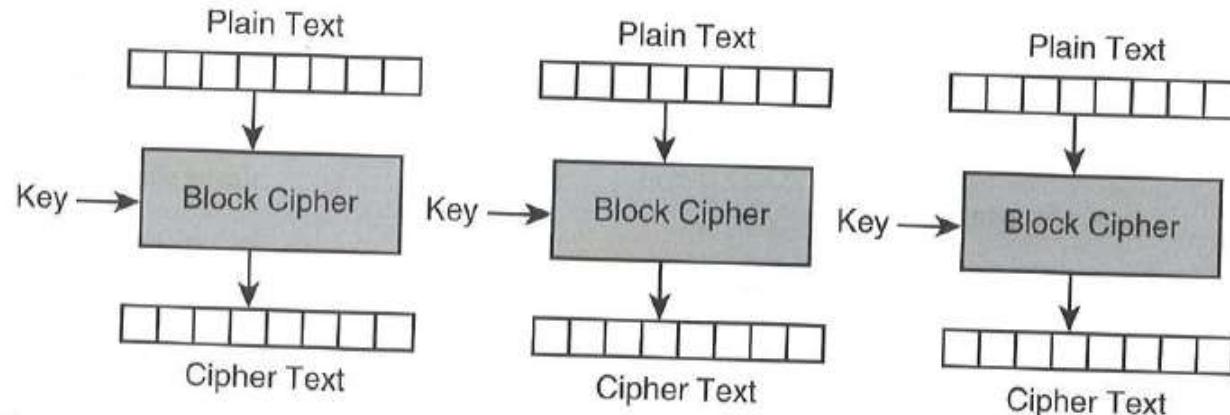
- **Biham and Shamir's Differential cryptanalysis in 1990**
 - A chosen plaintext attack
 - Required 2^{47} plaintext, ciphertext pairs to break the cipher
 - Observing strength change with change in algorithm through statistical analysis
 - Revealed several properties
 - Small changes in S-boxes weakened the cipher
 - Optimal design
 - Longer key size did not make it resistant to differential cryptanalysis
- **Matsui's Linear cryptanalysis in 1994**
 - Known plaintext attack
 - Relies on linear approximation and statistical analysis
 - Improved result
 - Required 2^{43} plaintext, ciphertext pairs

Single Execution DES (or other Block Cipher) Modes

- Direct mode or Electronic Code Book Mode
- Cipher Block Chaining Mode
- Cipher Feedback Mode
- Output Feedback Mode
- Counter

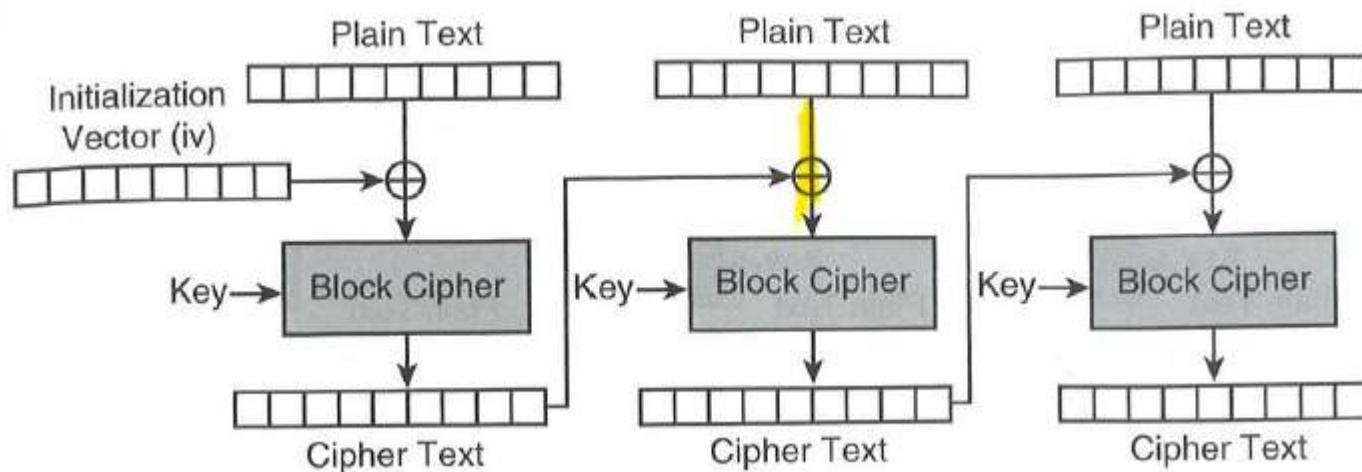
Electronic Code Book (ECB)

- Each block encrypted separately
 - Susceptible to attacks



Cipher Block Chaining (CBC)

- Ciphertext output from each block is used for the next block
 - XORed with next plaintext block before encryption
 - First block encrypted with IV (initialization vector)

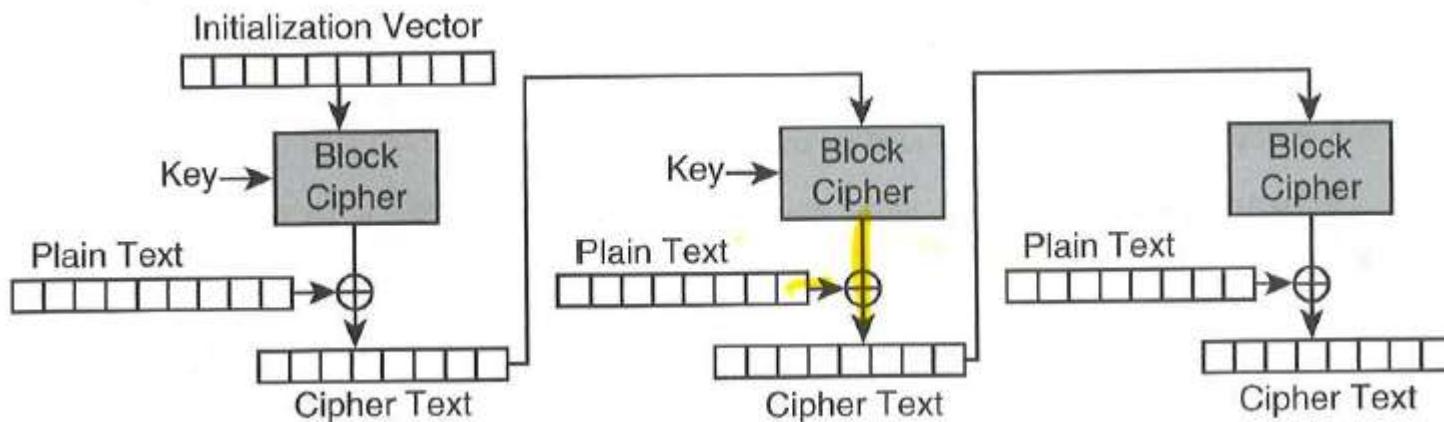


Self-Healing Property of CBC

- ❑ A.k.a. self-synchronizing..
- ❑ If one block of ciphertext is corrupted, error propagates to at most two blocks
 - I.e, if c_k is corrupted, it will only propagate up to c_{k+1} . And c_{k+2} would be error free.
 - Example:
 - Initial message
 - 3231343336353837 3231343336353837 3231343336353837 3231343336353837
 - Received as ciphertext (underlined 4c should be 4b)
 - ef7c4cb2b4ce6f3b f6266e3a97af0e2c 746ab9a6308f4256 33e60b451b09603d
 - Which decrypts to
 - efca61e19f4836f1 3231333336353837 3231343336353837 3231343336353837
 - Incorrect bytes underlined
 - Plaintext “heals” after 2 blocks

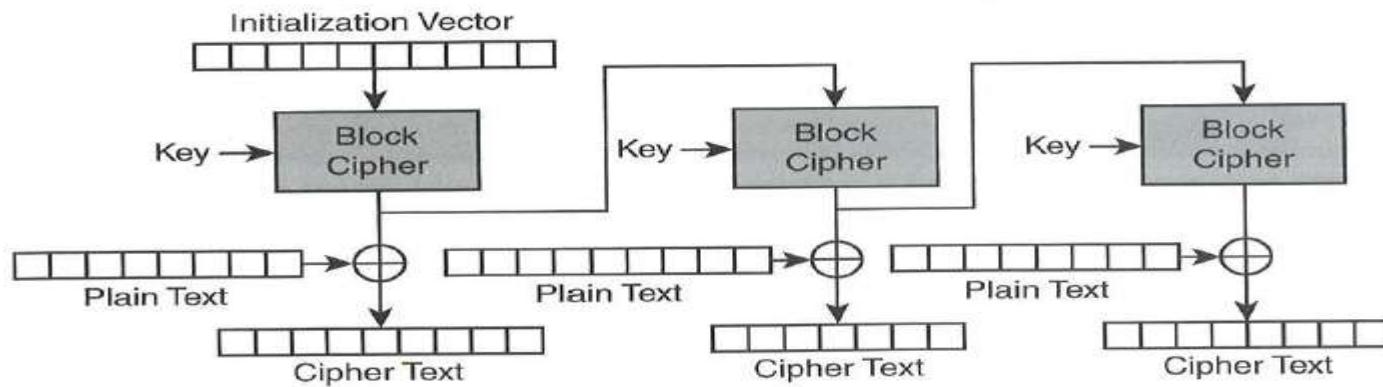
Cipher FeedBack (CFB)

- Ciphertext output from each block is used for the next block
 - Serves as the only input to encryption
 - Plaintext of next block is XORed with the encryption result to produce the next ciphertext



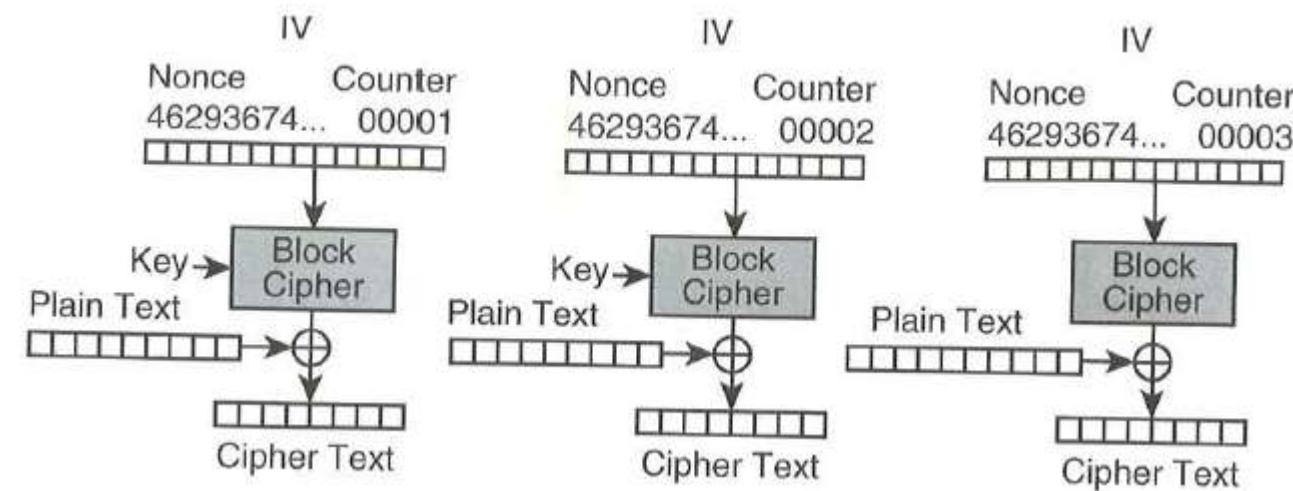
Output FeedBack (OFB)

- Output from each encrypted block is used for the next block
 - Serves as only input to encryption
 - Plaintext is XORed with the encryption result to produce the ciphertext



Counter (CTR)

- ❑ Uses a “nonce” (a random number that is used once) concatenated with a counter, which is encrypted by the block cipher, and the output XOR’d with the plaintext block to produce the ciphertext.



Mode Comparisons

	Input to Block Cipher	Cipher Text Generated by	Each block connected with previous block
ECB	Plain text	Output of block cipher	No
CBC	XOR of plain text and cipher text of previous block	Output of block cipher	Yes
CFB	Cipher Text of Previous Block	XOR of plain text and output of block cipher	Yes
OFB	Output of previous block cipher	XOR of plain text and output of block cipher	Yes
CTR	Random number	XOR of plain text and output of block cipher	No

Which Mode is Better?

□ Depends...

- Predictability
 - Do identical messages need to produce same cipher message or different one?
- Error propagation
 - Error prone medium?
- Efficiency
 - Need to be parallelizable?

Multiple Execution DES Modes

- Double execution (Double DES)**

- Triple execution (Triple DES)**

Double DES

- Encrypt-Encrypt Mode (2 keys: k, k')
 - $c = \text{DES}_k(\text{DES}_{k'}(m))$
- Double DES is only as secure as DES
 - Effective key length is same. Only doubles the work for attacker.
 - Susceptible to “meet in the middle attack”, a known plaintext attack on Double DES).

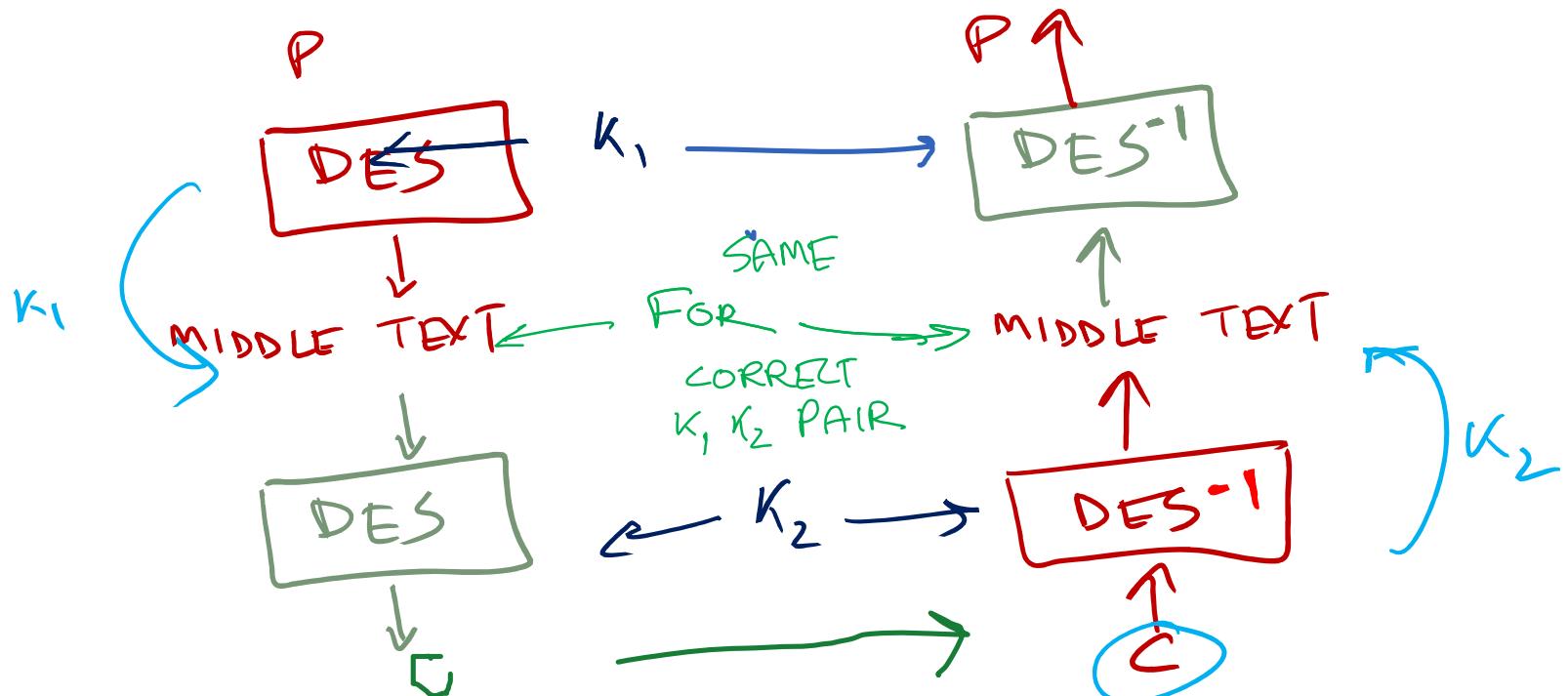
$$C = E_{k_2}(E_{k_1}(P))$$

$$M = E_{k_1}(P) = D_{k_2}(C)$$

$$P = D_{k_1}(D_{k_2}(C))$$

Meet in the Middle Attack

- ❑ A known plaintext attack where attacker does time memory tradeoff analysis while encrypting from one end and decrypting from other end and meeting in between where match is found to guess the keys



Triple DES

EDE MODE
→ MOST POPULAR

□ Modes:

- Encrypt-Decrypt-Encrypt Mode (2 keys: k, k')
 - $c = \text{DES}_k(\text{DES}_{k'}^{-1}(\text{DES}_k(m)))$
- Encrypt-Encrypt-Encrypt Mode (2 keys: k, k')
 - $c = \text{DES}_k(\text{DES}_{k'}(\text{DES}_k(m)))$
- Encrypt-Encrypt-Encrypt Mode (3 keys: k, k', k'')
 - $c = \text{DES}_k(\text{DES}_{k'}(\text{DES}_{k''}(m)))$

□ Triple DES much secure than DES

- **Double effective key length (112 bit). Still secure. Approved until 2030. Not suggested from 2005.**

DES to AES

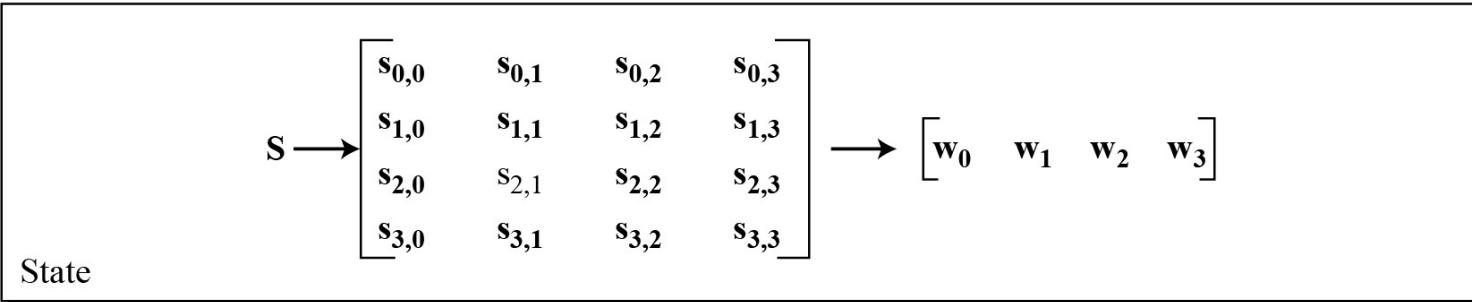
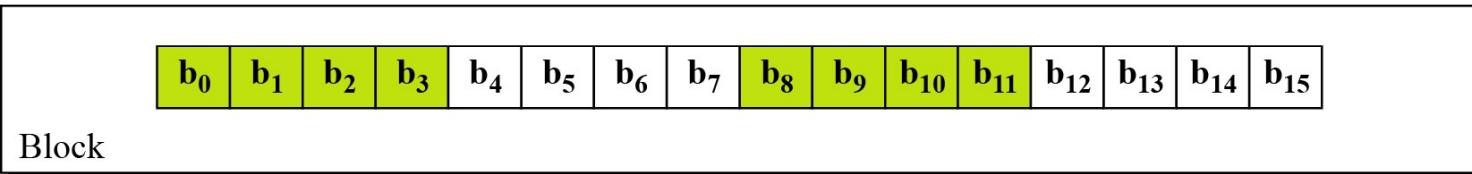
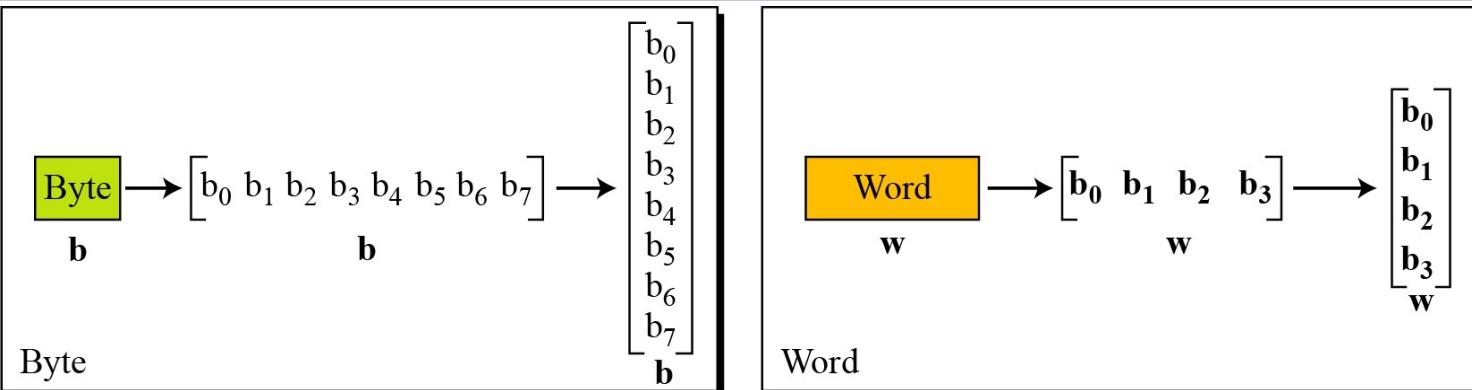
□ With future anticipation of technology,

- NIST called for contest in 1997**
- 5 finalists**
- Final selection based on security as well as cost and efficiency**
- NIST selected Rijndael from Dutch Cryptographers as winner**
- Advanced Encryption Standard became Federal Information Processing Standard, FIPS 197 (2001).**

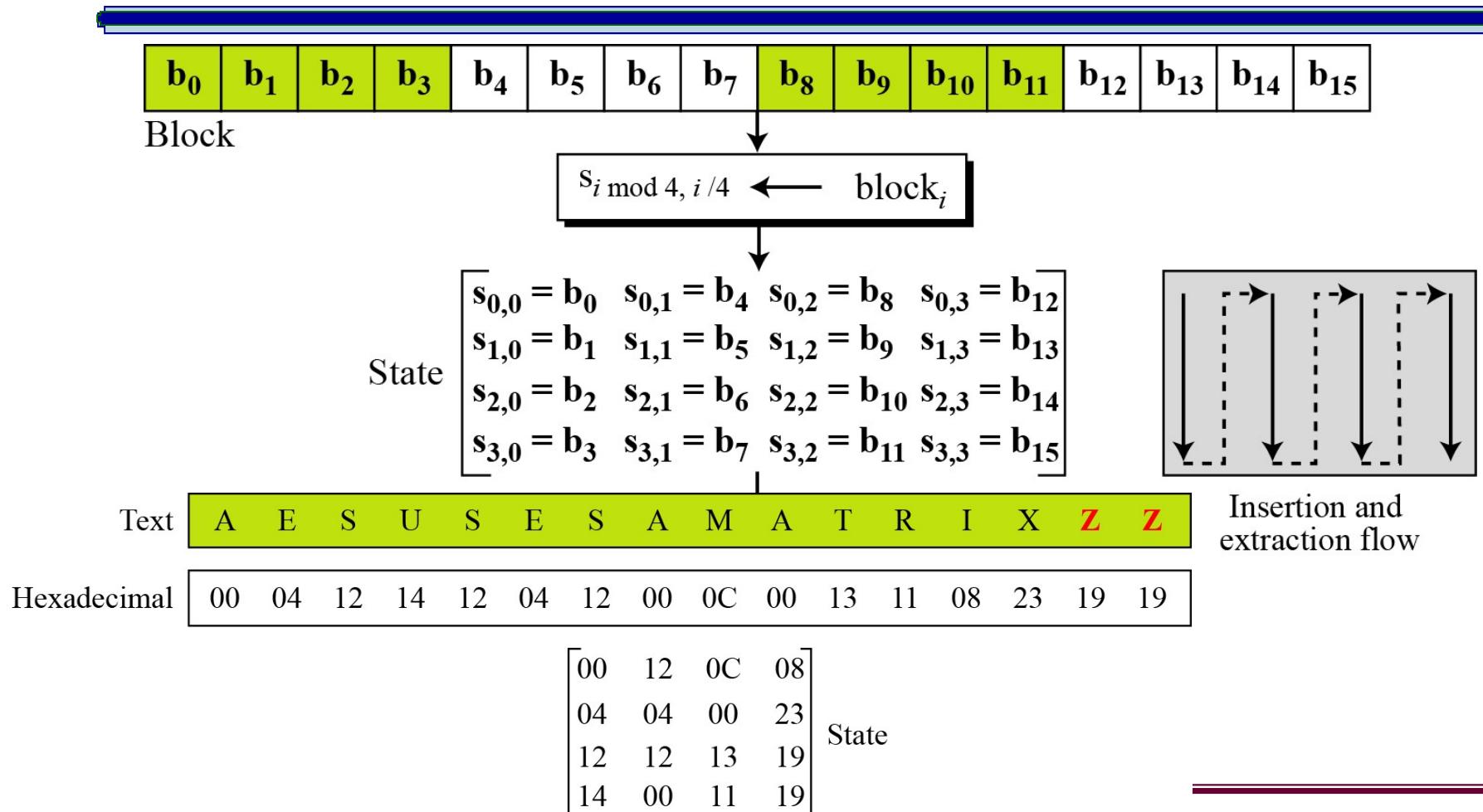
AES

- ❑ FIPS-197
- ❑ Designed to withstand attacks that were successful on DES
 - Security, cost, implementation
- ❑ Characteristics of AES
 - 128 bit block
 - Key size 128/192/256 bit and Rounds respectively 9/11/13
 - Uses bit level multiplication along with substitution and permutation
 - No Fiestal function
 - No secrecy

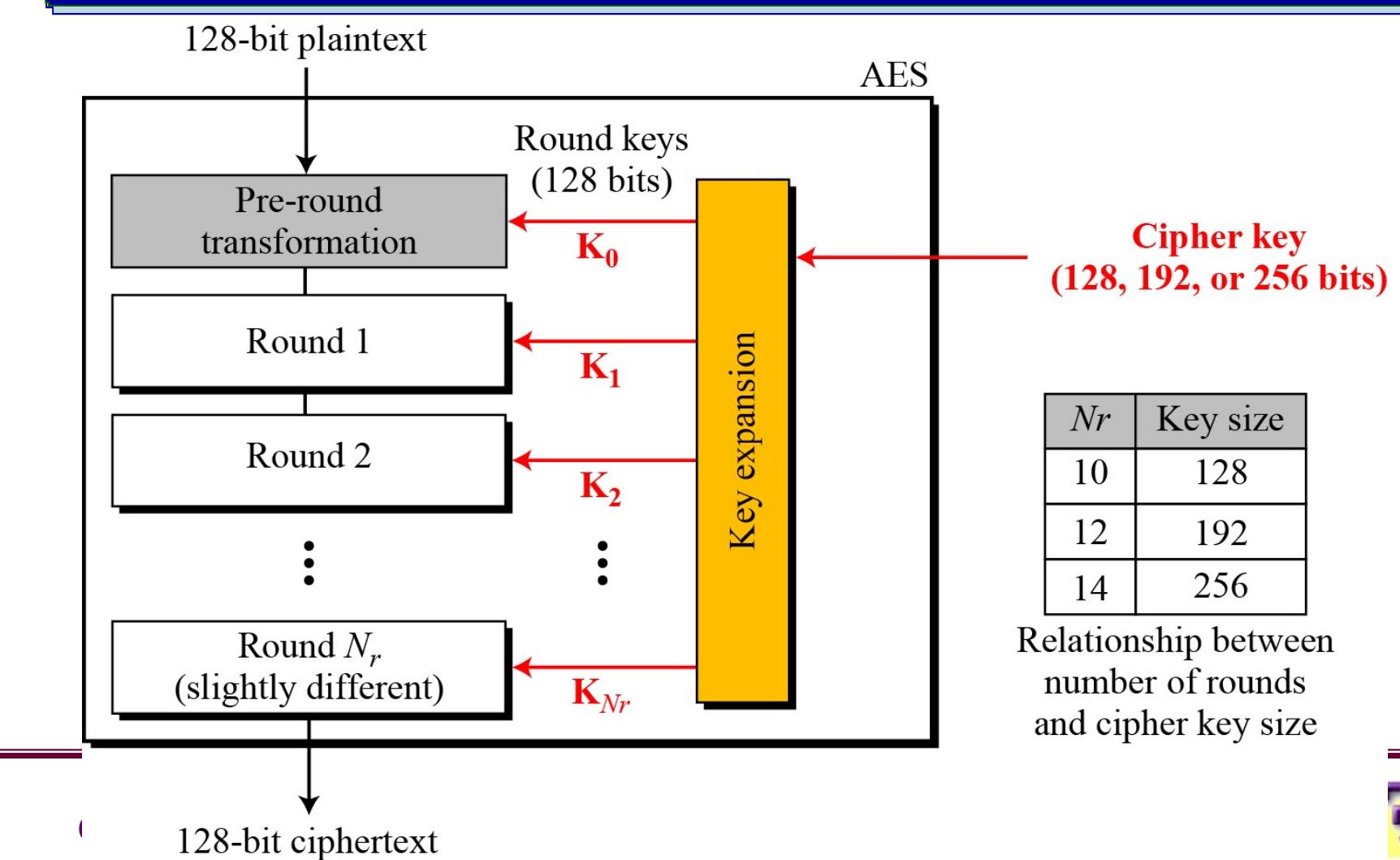
AES Data Units



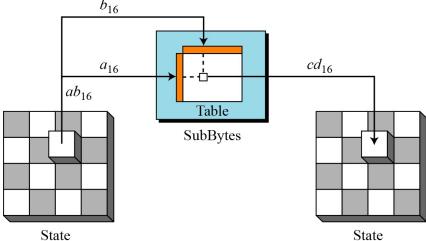
AES State



AES Design

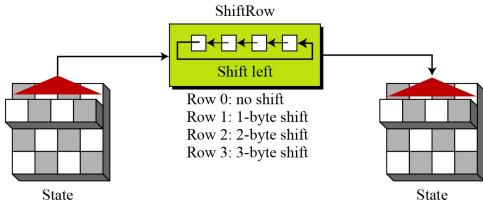


AES Round

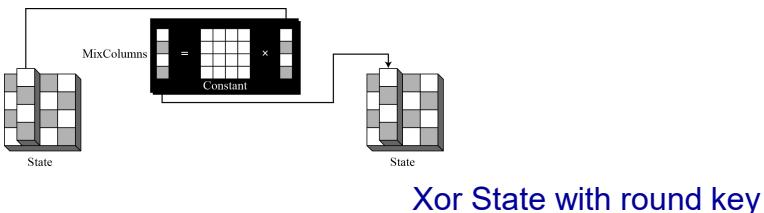


Substitutes each byte in State with another byte using a lookup table with row and column index

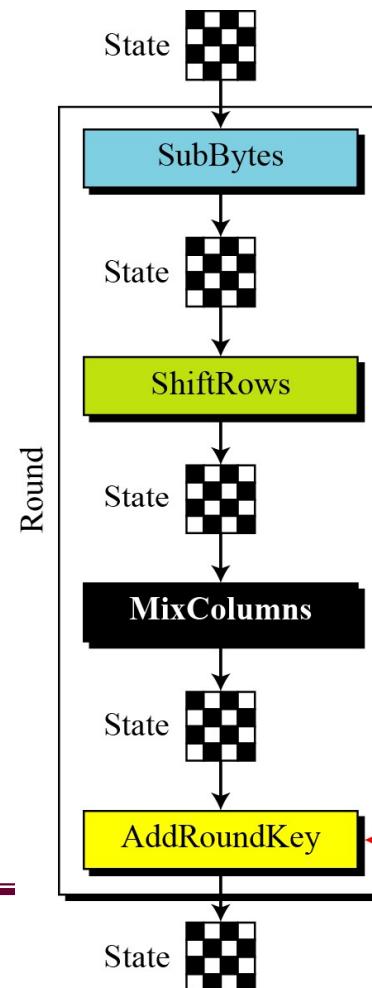
Shifts bytes in the State rows cyclically



Transforms each State column



Xor State with round key



Notes:

1. One AddRoundKey is applied before the first round.
2. The third transformation is missing in the last round.

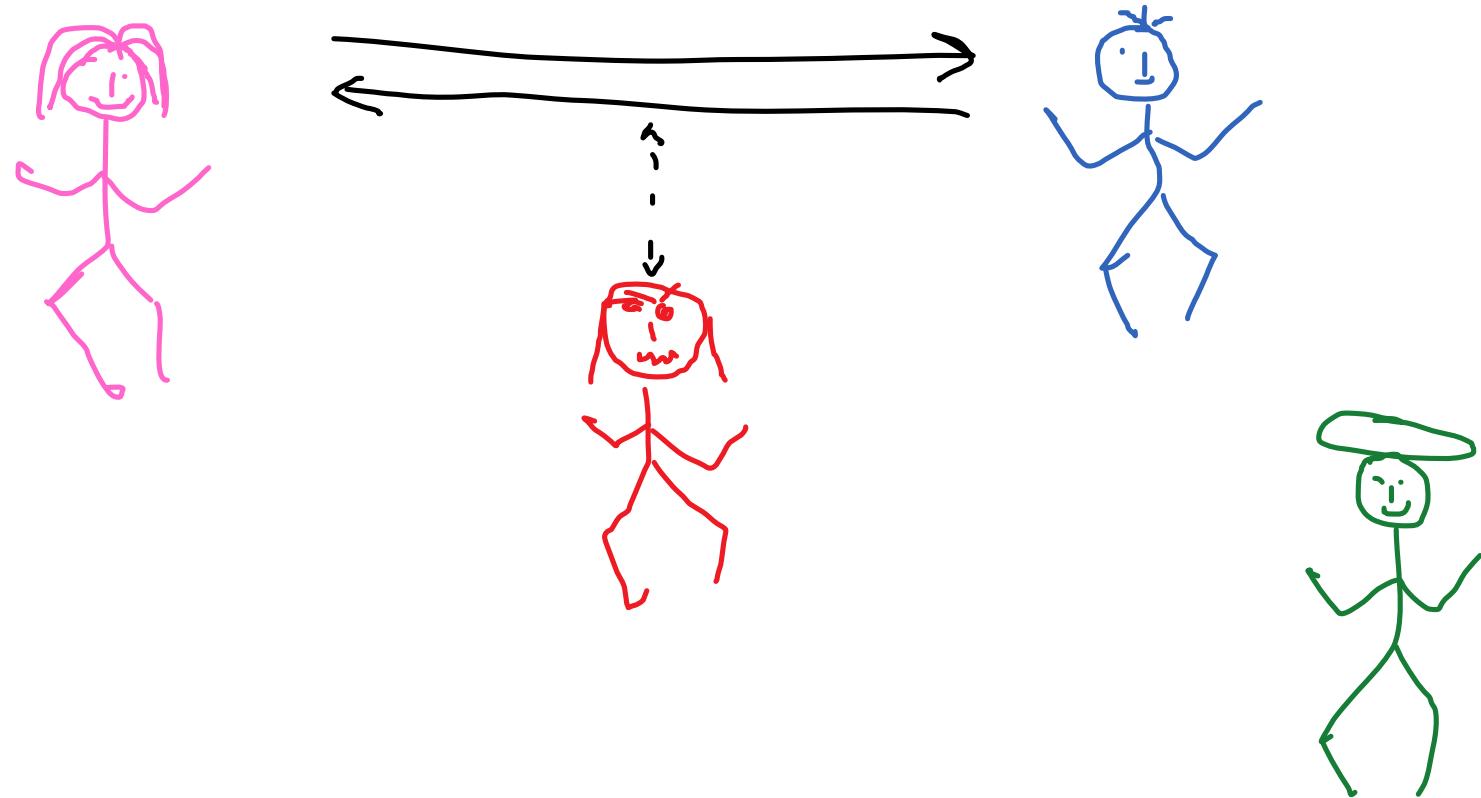
AES Security

- Has not been broken “yet”!
 - No brute force attacks
 - No statistical attacks
 - No linear/differential attacks
- Efficient and simple implementation
 - Software, hardware, firmware
- Available royalty free world wide
- Check this out:
<http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html>

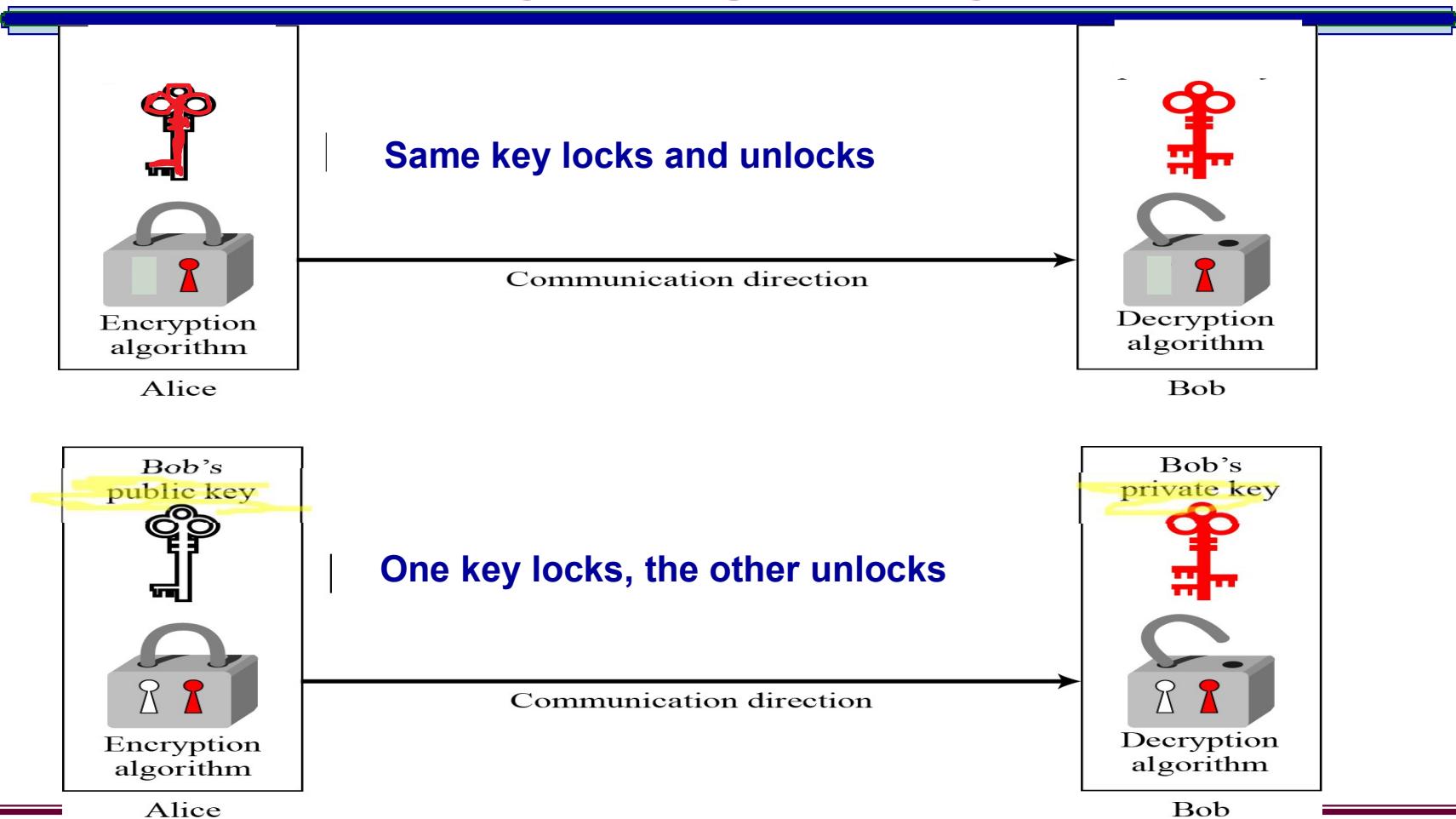
Public Key Cryptography

- Asymmetric key cryptography
- Proposed by Diffie and Hellman, 1976
- Two keys
 - *Private key known only to individual*
 - *Public key available to anyone*

Meet Alice, Bob and Eve (and Cathy)

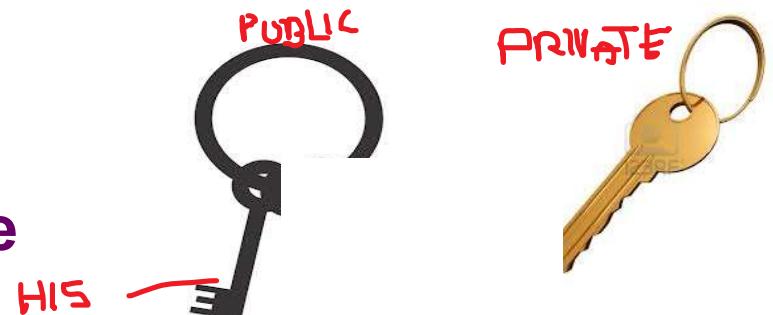


Symmetric and Asymmetric Cryptography



Public and Private Key

- Mathematically related AND generated together
 - If message encrypted by one, can only be decrypted by the other
- For example, 3 users
 - Moe, Larry and Curly
- Each of the three has ONE pair of keys
 - Public key
 - Known/accessible to anyone who wants to communicate with that user
 - Private key
 - ONLY known to that user
- Each of three would also have
 - Public key of the other users



Public Key Cryptography Goal

- Confidentiality
- Data authentication (Integrity)
- Origin authentication
 - Non-repudiation

Requirements

1. It must be computationally easy to encipher or decipher a message given the appropriate key
2. It must be computationally infeasible to derive the private key
 - from the public key
 - from a chosen plaintext attack

RSA

- Rivest, Shamir, Adleman, 1977
- Exponentiation cipher
 - Based on exponentiation arithmetic

Background

□ One way function properties:

- Given x , $y=f(x)$ is easily computed.
- Given y , $x=f^{-1}(y)$ is computationally infeasible to calculate.

EXPOENENTIATION : $y = a^x$

INVERSE EXPONENTIATION / LOGARITHM :
 $x = \log_a y$

□ Trapdoor: One way function with additional property:

- Given y and a secret (trapdoor), x can be computed easily.

Background cont.

□ Prime numbers

- Numbers that have no factor common (or GCD=1) with any other numbers

□ Totient function $\phi(n)$

- The **number** of numbers relatively prime/coprime (no factors in common or GCD=1) to a **larger integer n**

□ Example: $\phi(10) = 4$

- 1, 3, 7, 9 are relatively prime to 10

□ Example: $\phi(21) = 12$

- 1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20 are relatively prime to 21

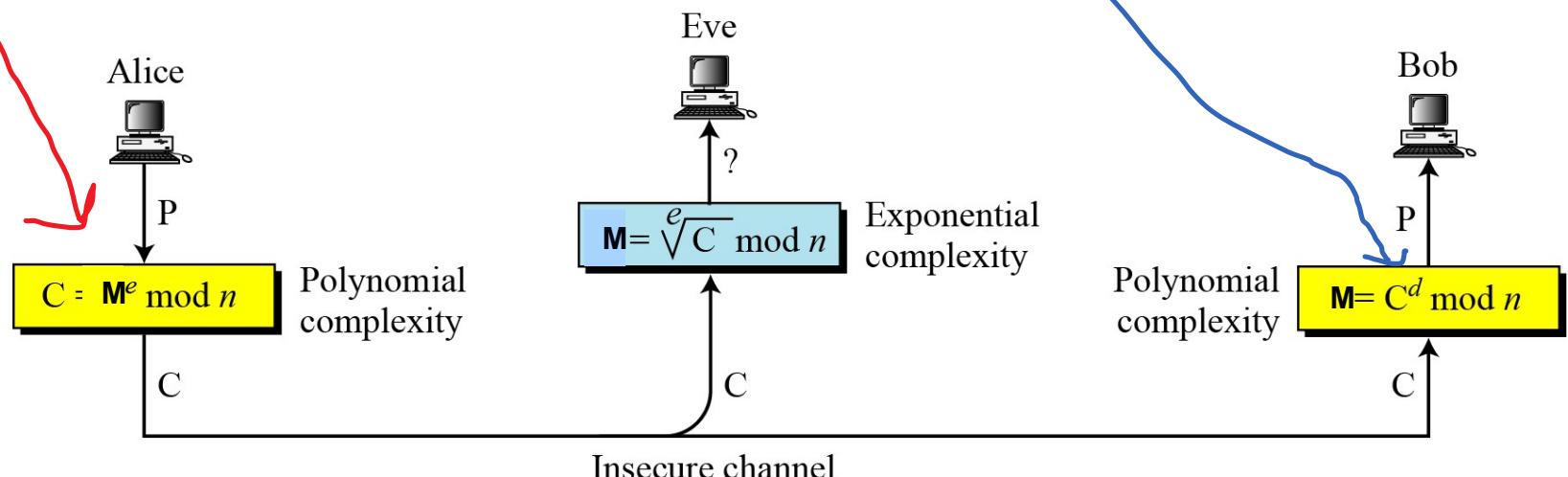
RSA Algorithm

- Choose two large prime numbers p, q
 - Compute $n = pq$
 - Compute $\phi(n) = (p-1)(q-1)$
 - Choose $1 < e < n$ such that e is relatively prime to $\phi(n)$.
 - Compute d such that $ed \bmod \phi(n) = 1$
- Public key: (e, n) ; private key: d
- Encipher: $c = m^e \bmod n$
- Decipher: $m = c^d \bmod n$

$m^e \rightarrow m \times m \times m \dots$
 e times

One Way Function and Trapdoor in RSA

- If attackers intercepts cipher text, recovering plain text without knowing private key, is infeasible.



Strength of RSA

❑ Relies on the difficulty of

- Finding factors to a large number
- Reversing the exponentiation function

❑ $ed \bmod \phi(n) = 1$

$n=pq$

$$\frac{ed - 1}{\phi(n)} = K \text{ SOME CONSTANT}$$

$\phi(n)$ is equal to $(p-1)(q-1)$

❑ Knowing $e \& n$ does not make deducing d trivial, because attacker

- won't know K
- won't know $p \& q$ PAIR

~~Confidentiality with RSA~~

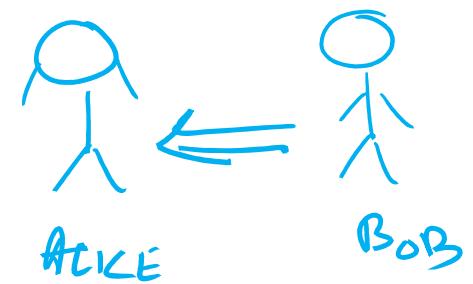
- To send message**
 - Encipher using receiver's public key
 - Decipher using receiver's private key

Example: Confidentiality with RSA

- Take $p = 7$, $q = 11$, so $n = 77$
- Alice is receiver and Bob is sender
- Alice chooses $e = 17$, making $d = 53$
- Bob wants to send Alice secret message
HELLO (07 04 11 11 14)
 - $07^{17} \text{ mod } 77 = 28$
 - $04^{17} \text{ mod } 77 = 16$
 - $11^{17} \text{ mod } 77 = 44$
 - $11^{17} \text{ mod } 77 = 44$
 - $14^{17} \text{ mod } 77 = 42$

$m^e \bmod n$

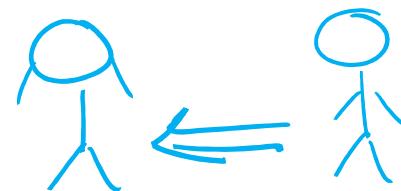
Alice's keys: public (17, 77); private: 53
Bob's keys: public: (37, 77); private: 13



Example cont.

- Alice receives 28 16 44 44 42
- Alice uses private key, $d = 53$, to decrypt message:
 - $28^{53} \text{ mod } 77 = 07$ H
 - $16^{53} \text{ mod } 77 = 04$ E
 - $44^{53} \text{ mod } 77 = 11$ L
 - $44^{53} \text{ mod } 77 = 11$ L
 - $42^{53} \text{ mod } 77 = 14$ O
- Alice translates message to letters to read HELLO

Alice's keys: public (17, 77); private 53
Bob's keys: public: (37, 77); private: 13



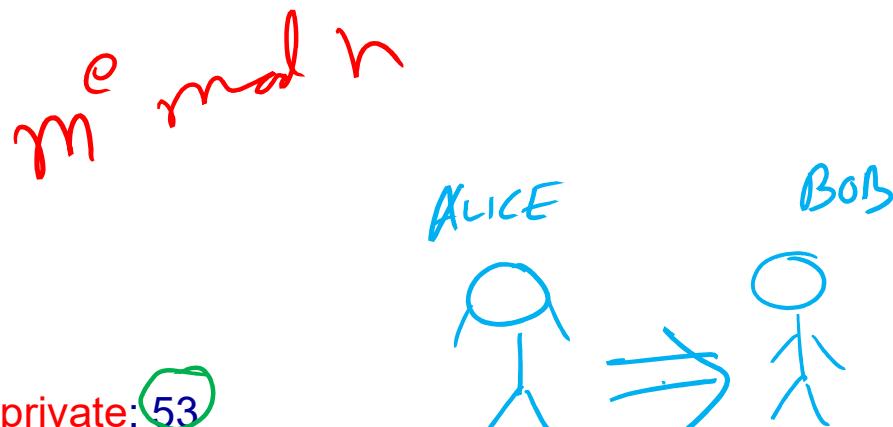
✓ Integrity with RSA

□ To send message

- Encipher using sender's private key
- Decipher using sender's public key

Example: Data and Origin Authentication with RSA

- ❑ Alice is sender and Bob is receiver
- ❑ Alice wants to send Bob message HELLO (07 04 11 11 14) so Bob knows it is what Alice sent (no changes in transit, and authenticated)
 - $07^{53} \text{ mod } 77 = 35$
 - $04^{53} \text{ mod } 77 = 09$
 - $11^{53} \text{ mod } 77 = 44$
 - $11^{53} \text{ mod } 77 = 44$
 - $14^{53} \text{ mod } 77 = 49$

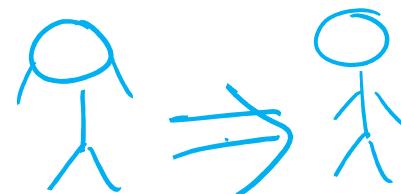


Example cont.

- Bob receives 35 09 44 44 49
- Bob uses Alice's public key, $e = 17$, $n = 77$, to decrypt message:
 - $35^{17} \text{ mod } 77 = 07$
 - $09^{17} \text{ mod } 77 = 04$
 - $44^{17} \text{ mod } 77 = 11$
 - $44^{17} \text{ mod } 77 = 11$
 - $49^{17} \text{ mod } 77 = 14$
- Bob translates message to letters to read HELLO
 - If (enciphered) message's blocks (letters) altered in transit, would not decrypt properly

Alice's keys: public (17, 77); private: 53
Bob's keys: public: (37, 77); private: 13

c m a g h n



✓ Confidentiality and Integrity with RSA

- Double encipherment on sender's part
 - First encipher using sender's private key
 - and then encipher the result again with receiver's public key

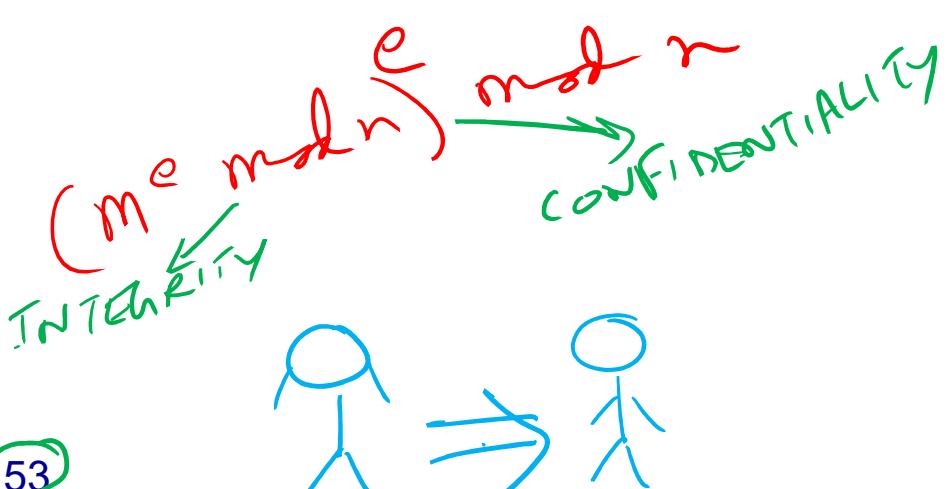
- Double decipherment on receiver's part
 - First decipher using receiver's private key
 - and then decipher the result again with sender's public key

Example: Confidentiality, Data and Origin Authentication with RSA

- Alice is sender and Bob is receiver
- Alice wants to send Bob message HELL both confidentiality and integrity-checked
- Alice enciphers HELL (07 04 11 11):
 - $(07^{53} \text{ mod } 77)^{37} \text{ mod } 77 = 07$
 - $(04^{53} \text{ mod } 77)^{37} \text{ mod } 77 = 37$
 - $(11^{53} \text{ mod } 77)^{37} \text{ mod } 77 = 44$
 - $(11^{53} \text{ mod } 77)^{37} \text{ mod } 77 = 44$

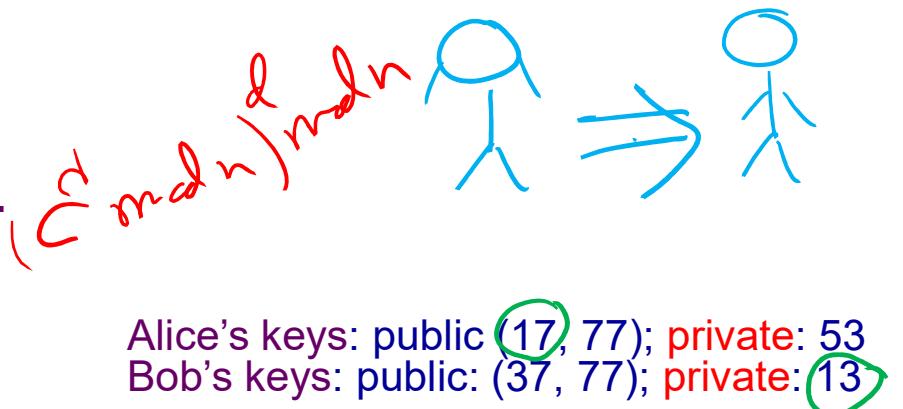
1st 2nd

Alice's keys: public (17, 77); private: 53
Bob's keys: public: (37, 77); private: 13



Example cont.

- Bob receives message from Alice
 - Alice's keys: public (17, 77); private: 53
 - Bob's keys: public: (37, 77); private: 13
- Bob deciphers 07 37 44 44 14:
 - $(07^{13} \text{ mod } 77)^{17} \text{ mod } 77 = 07$
 - $(37^{13} \text{ mod } 77)^{17} \text{ mod } 77 = 04$
 - $(44^{13} \text{ mod } 77)^{17} \text{ mod } 77 = 11$
 - $(44^{13} \text{ mod } 77)^{17} \text{ mod } 77 = 11$
- Bob received 07 04 11 11 or HELL



Security Services by RSA

Confidentiality

- Text enciphered with public key cannot be read by anyone except the owner of the private key

Data Integrity

- Enciphered letters cannot be changed undetectably without knowing private key

Origin authentication

- Only the owner of the private key knows it, so text enciphered with private key must have been generated by the owner
- Non-repudiation
 - Message enciphered with private key came from someone who knew it

Attacks against RSA

- Although RSA key is secure enough, susceptible to inference attacks similar to substitution ciphers
- If 1 character per block
 - Statistical attacks to guess plain text
 - Reordering of letters attack to alter message meaning
 - Example: reverse enciphered message of text ON to get NO or text LIVE to get EVIL
 - Padding helps where block identification is sent along

1024 BIT BLOCKS → 6 BIT BLOCK NUMBER + 8 BIT CHARACTER
+ 1010 BIT RANDOM DATA

- If possible plaintext is known, Forward searching or precomputation attack to guess plain text sent
 - Example: Precompute Buy or Sell ciphertext and then match with transmission
 - Padding helps where random data makes precomputation difficult

Use of RSA

- Still widely used in e-commerce
- Secure with
 - long keys *(e d n)* $2^{16} + 1$
 - Typically large public *e* (65537 – mandated by NIST)
 - n usually 1024-2048 bits long, p and q usually at least 512 digits
 - p and q not “too close”

□ Interesting link

<http://www.crypto.org/images/ct1/presentations/RSA/RSA-Flash-en/player.html>

Comparison between Symmetric and Asymmetric

BASED ON SHARED SECRET

Symmetric

- One secret key
- Less complex computation
- Easier implementation
- Faster
- Harder distribution

n
PEOPLE

$$\frac{n(n-1)}{2} \text{ sym key}$$

- No non-repudiation
- Used for general purposes

BASED ON PERSONAL SECRET

Asymmetric

- Two keys (one secret and one public)
- More complex computation
- More complex implementation
- Slower
- Easier distribution

n Assym. KEY

- Non-repudiation
- Used for specialized purposes

Checksum

- Digital checksums are compact representation of data
 - Provides integrity service
- Simple checksum uses 1 unit of data to detect errors in rest
 - Use of ASCII parity bit
 - ASCII has 7 bits; 8th bit is “parity” (can be 0/1)
 - Types
 - Even parity: parity bit set to get even number of 1 bits
 - Odd parity: parity bit set to get odd number of 1 bits
 - Bob receives “10111101” as bits.

PPR
B

Cryptographic Checksum/Hash

- Function/method to generate a set of k bits from a set of n bits (where $k \leq n$).
 - For any given message, easy to compute the hash (not vice versa)
 - For a given message, the corresponding hash can not be predicted (only computed to be seen)
 - Same message will produce the same hash
 - Any change in message will generate completely different hash
 - A.k.a.,
 - Message digest
 - Message Integrity Code (MIC)
 - Modification Detection Code (MDC)
-

Hash is NOT Encryption

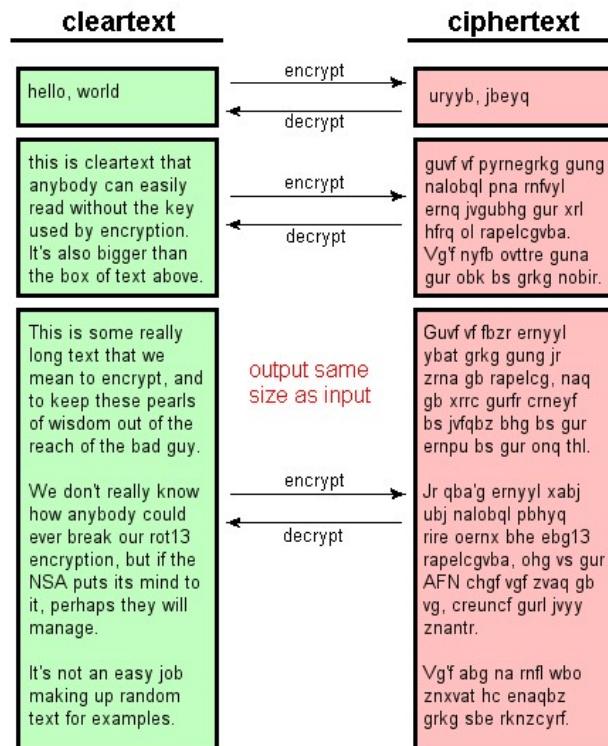


Fig. 1: Encryption - a two-way operation

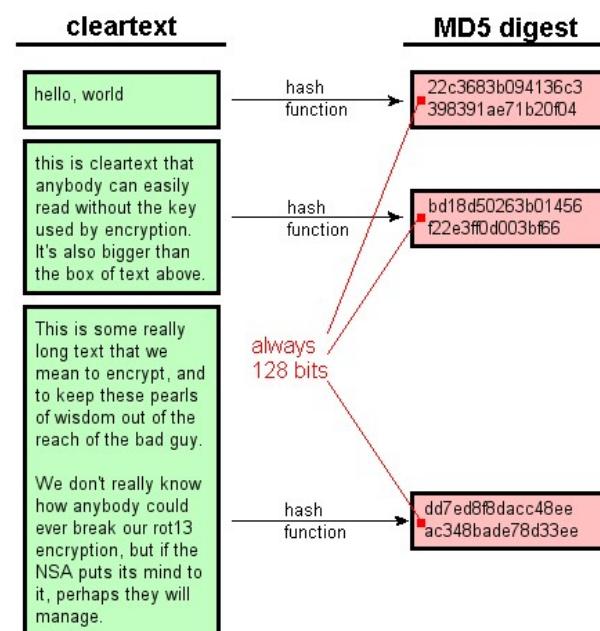


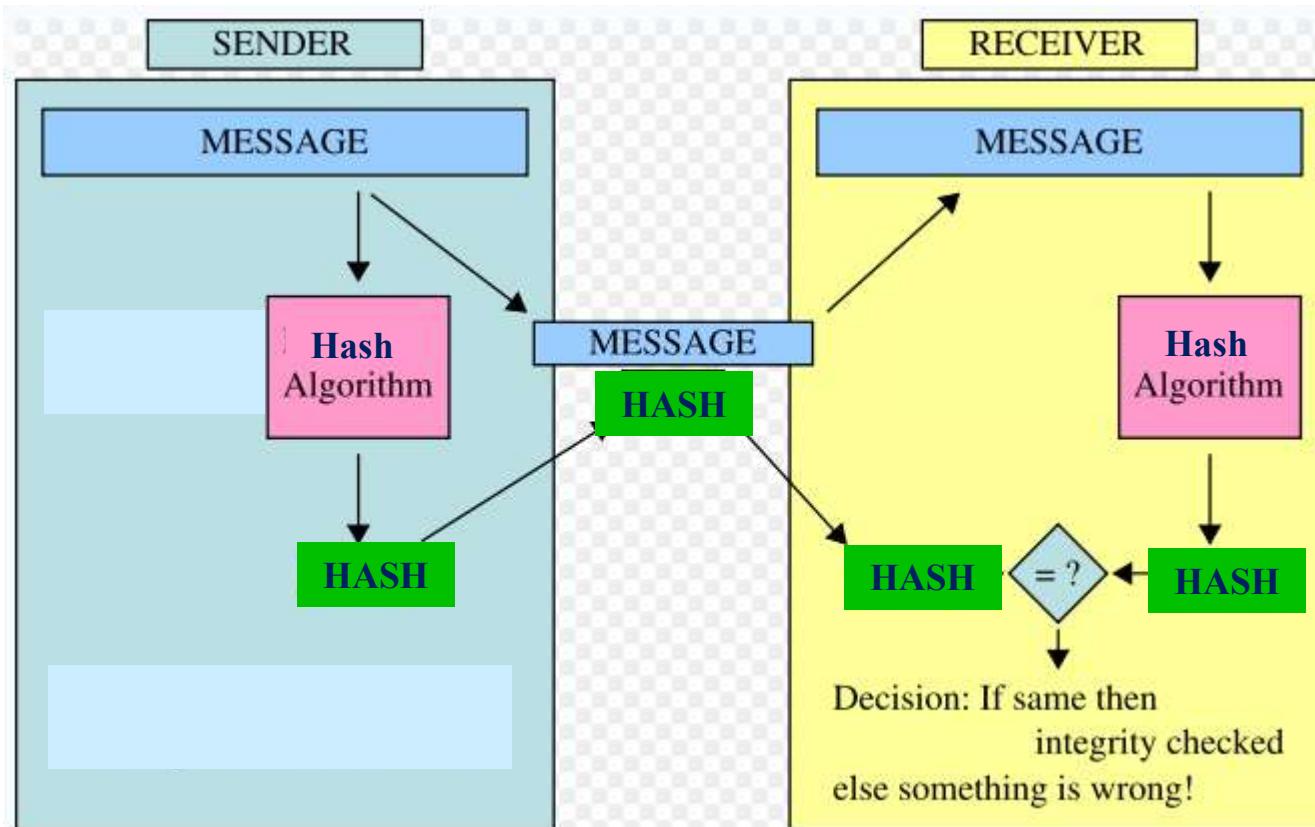
Fig. 2: Hashing - a one-way operation

<http://www.unixwiz.net/techtips/iguide-crypto-hashes.html>

Cryptographic Hash Properties

- Impossible to derive original message from digest
 - Fixed-length digest regardless of message size
 - A digest should be result of the whole message, not a portion of it
-

Checksum/Hash

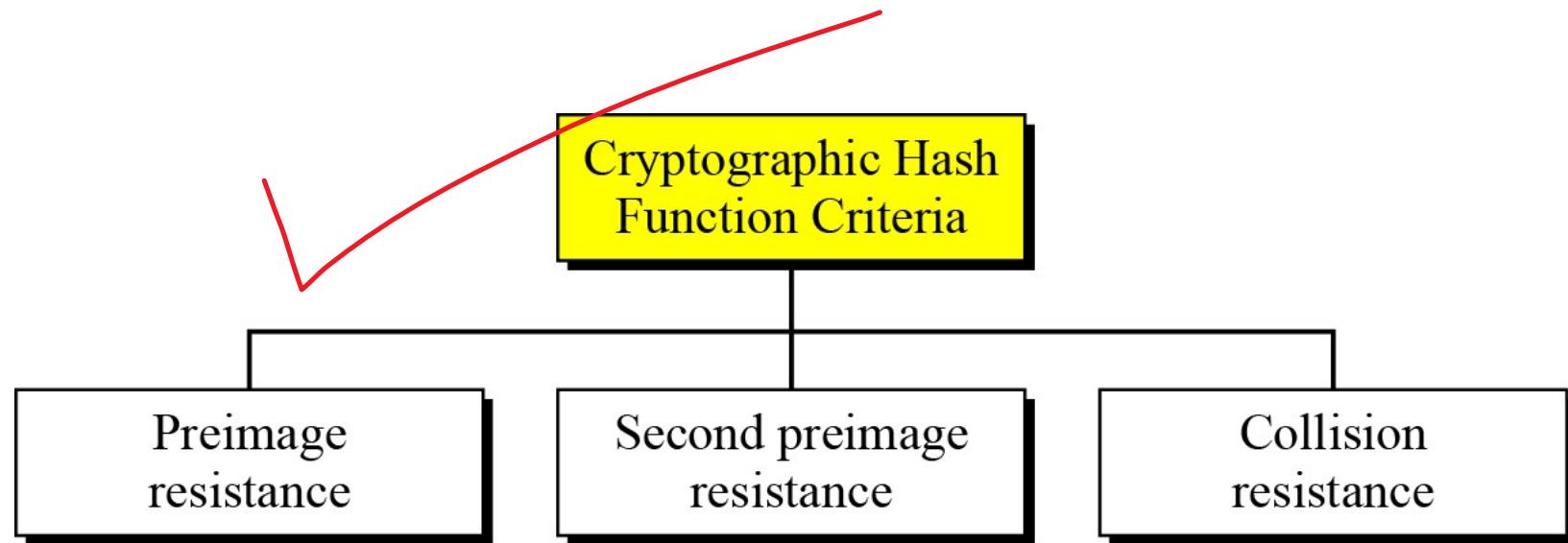


Adapted from <http://en.wikipedia.org/wiki/Image:MAC.svg>

Applications of Hash

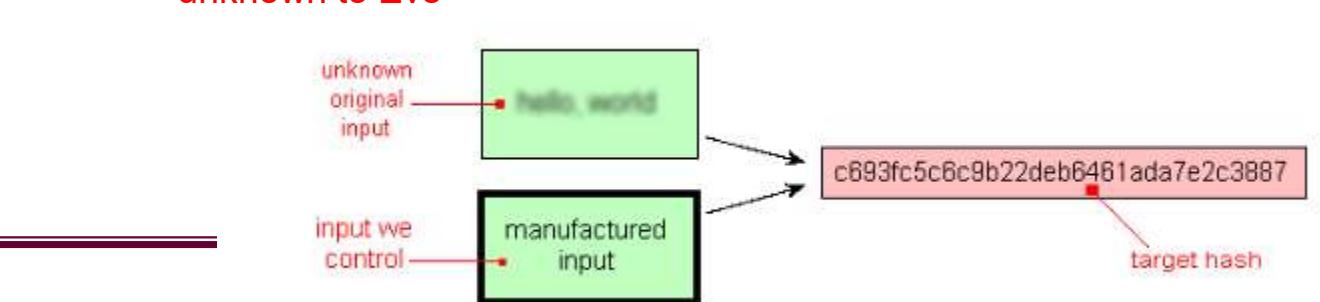
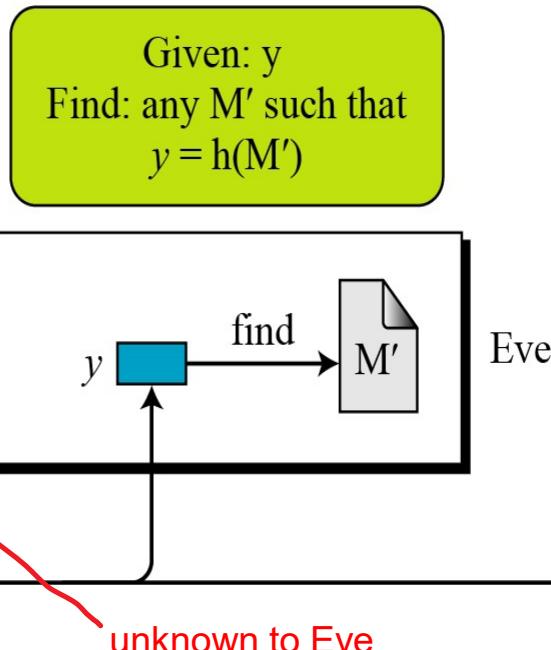
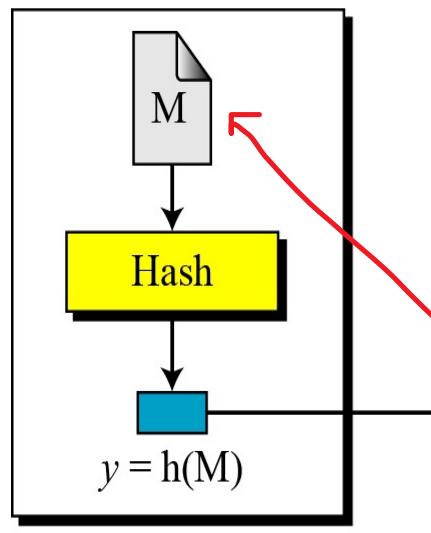
- Message (in communication) integrity
 - Document/file (in storage) integrity
 - Application/system state integrity
 - Identifier of entity
 - Key generation
-

Resistance Criteria of Cryptographic Hash Function

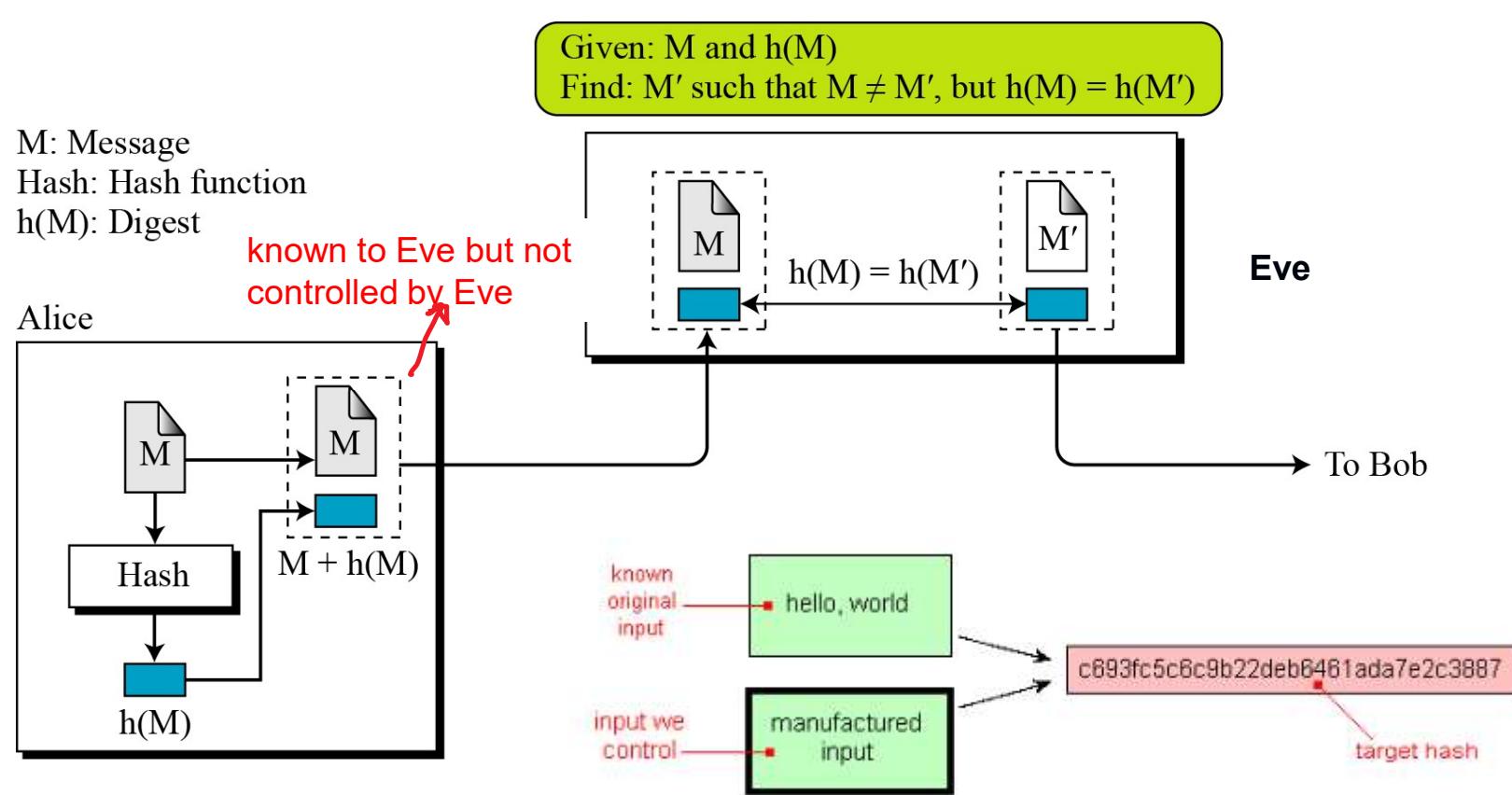


Preimage Resistance

M: Message
Hash: Hash function
 $h(M)$: Digest



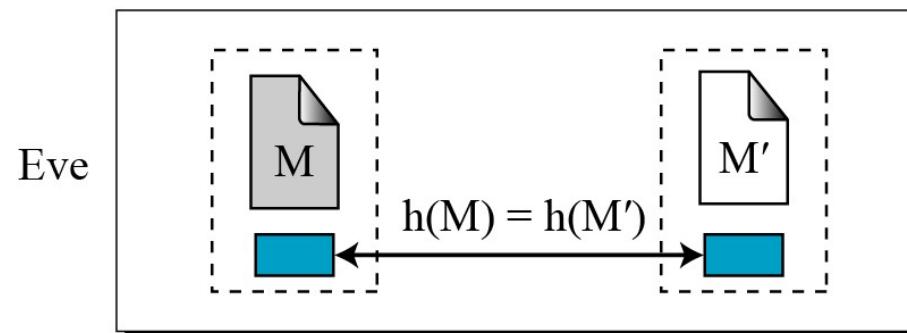
Second Preimage



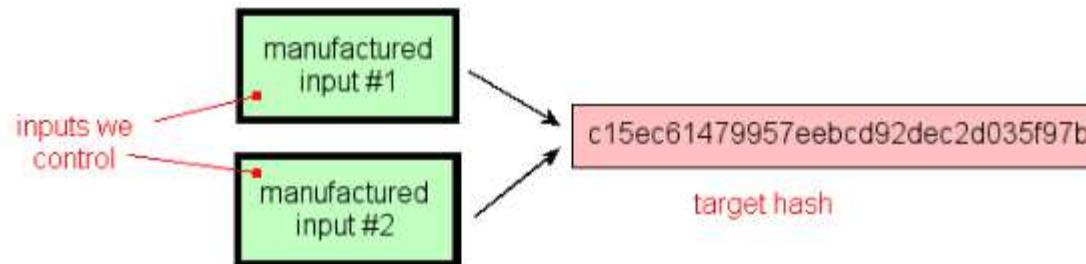
Collision Resistance

M: Message
Hash: Hash function
h(M): Digest

Find: M and M' such that $M \neq M'$, but $h(M) = h(M')$



Both controlled by Eve



Summary of Hash Resistance

- No other message should produce the same digest
 - Given only hash/digest PR
 - Given only hash/digest and original message SR
- No two messages should result in the same hash/digest CR

Collisions

- When two different messages have same hash, they are called collisions.

If $x \neq x'$ and $h(x) = h(x')$, x and x' are a *collision*

- Pigeonhole principle: if there are n containers for $n+1$ objects, then at least one container will have 2 objects in it.
 - Example: if there are 32 files and 8 possible cryptographic checksum values, at least one value corresponds to at least 4 files

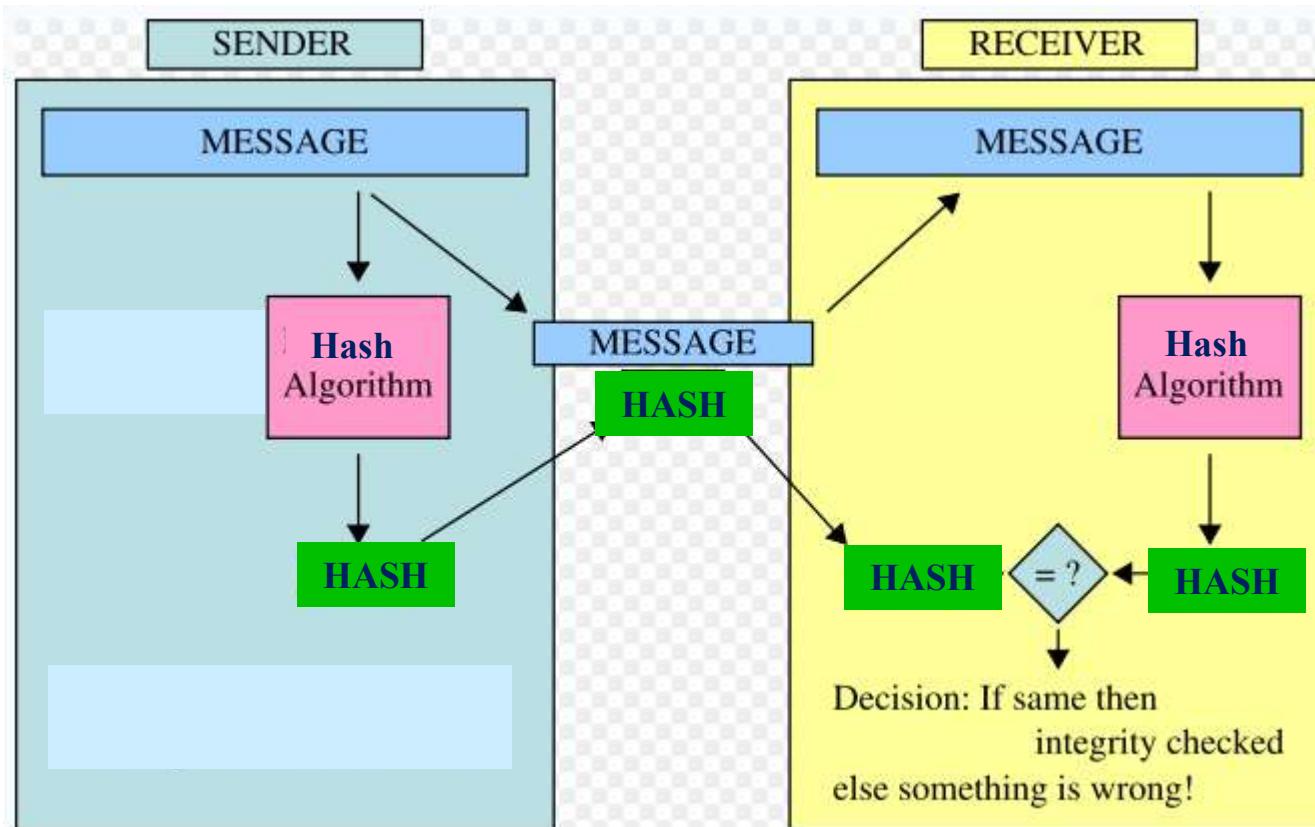
- Birthday Attack

- Used for finding collisions to commit forgery
- Possible because of birthday paradox
 - Probability that from a group of random number of people at least a pair will have the same birthday
 - The probability increases with pigeonhole principle, for example, this probability will be maximum when there is more than 366 people in the group

Key use in checksums

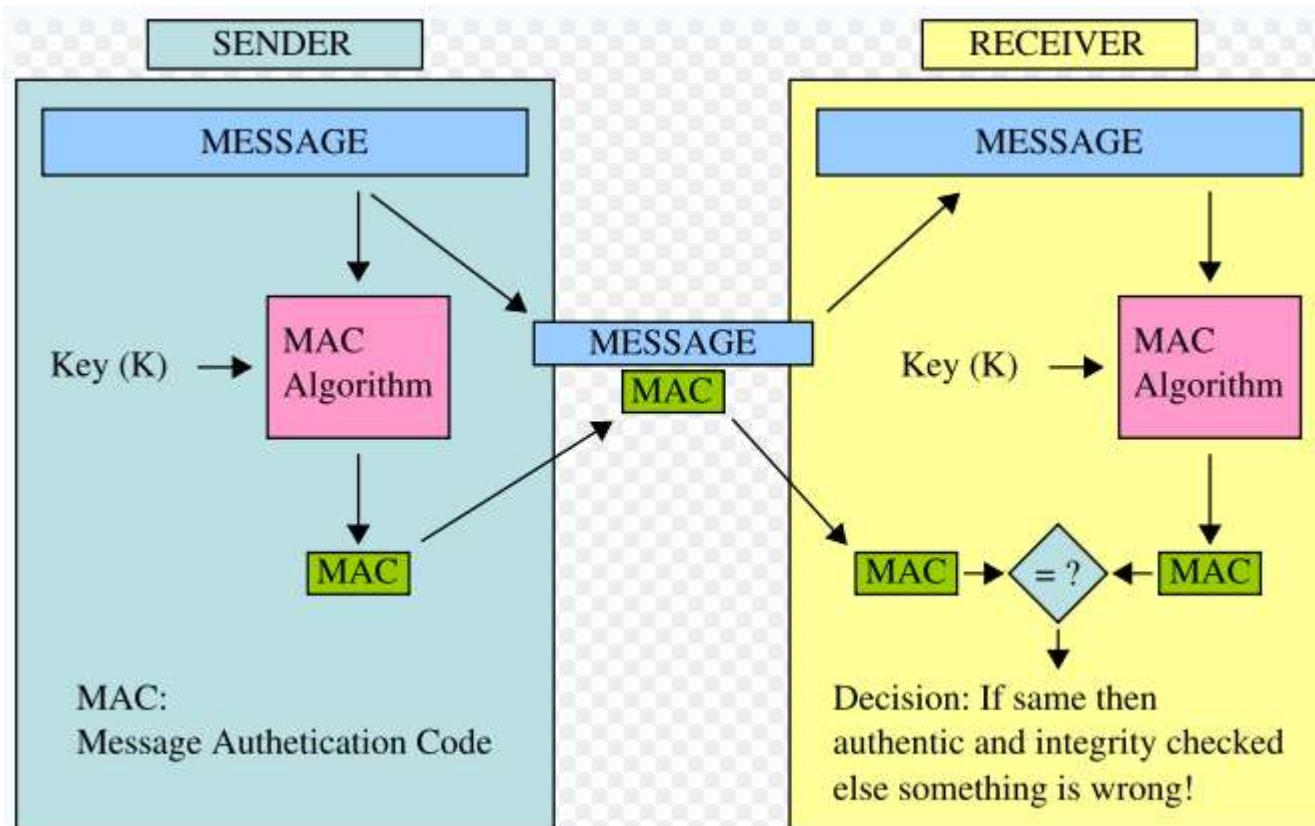
- ❑ Keyless cryptographic checksum: requires no cryptographic key
 - MD5 and SHA-1 are best known; others include MD4, HAVAL, and Snefru
 - Susceptible to forgery
- ❑ Keyed cryptographic checksum: requires cryptographic key
 - Can be used for message authentication (origin integrity)
 - Called Message Authentication Code (MAC)
 - Not same as Digital Signatures
- ❑ Keyed checksum provides more assurance than keyless

Checksum/Hash



Adapted from <http://en.wikipedia.org/wiki/Image:MAC.svg>

MAC



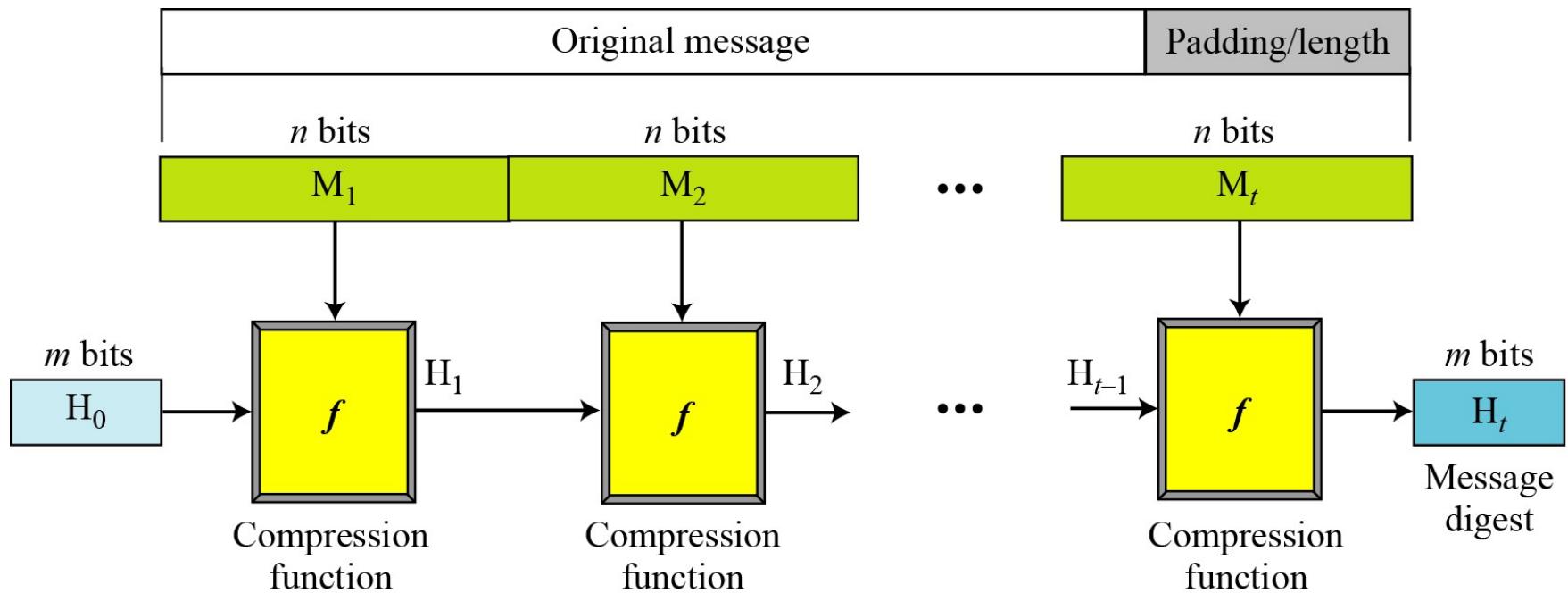
<http://en.wikipedia.org/wiki/Image:MAC.svg>

Use of Key in MAC

- Usually the key is concatenated with the message and then hashed to produce MAC
 - Prefix, postfix or both
 - Subject to brute force attack for key, unless size of key makes it difficult
 - Strength of MAC depends on strength of hash algorithm used
 - To improve security, iterated or nested MACs are used
-

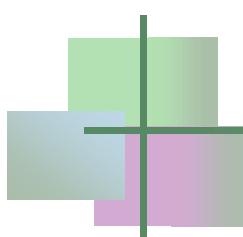
Iterated Hash with Compression Function

Merkle-Damgård Scheme



Compression (Hash) Functions

- Generates an m-bit string from a string of n-bit where $m \leq n$
- Two types
 - Made from Scratch
 - MD5, SHA
 - Based on block ciphers
 - Whirlpool



SHA Features

Table 12.1 Characteristics of Secure Hash Algorithms (SHAs)

Characteristics	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Maximum Message size	$2^{64} - 1$	$2^{64} - 1$	$2^{64} - 1$	$2^{128} - 1$	$2^{128} - 1$
Block size	512	512	512	1024	1024
Message digest size	160	224	256	384	512
Number of rounds	80	64	64	80	80
Word size	32	32	32	64	64

~~CBC-MAC or CMAC~~ ✓(Hash with block cipher)

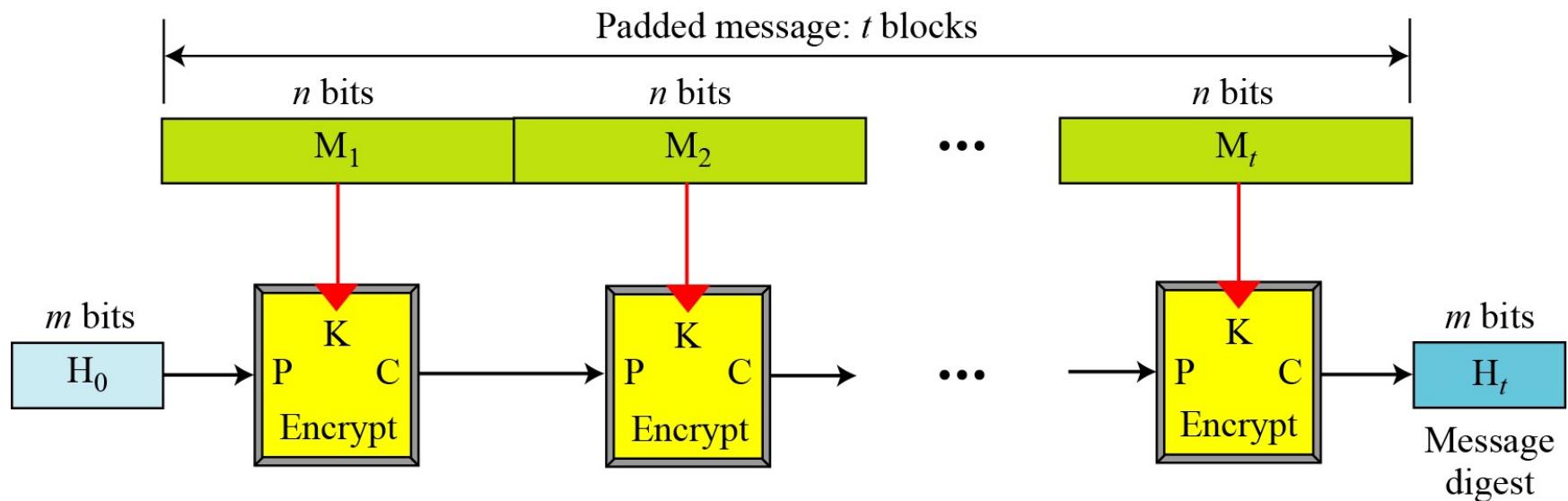
□ DES in CBC mode

- Encipher message, use last n bits as checksum.
 - Requires a key to encipher, so it is a keyed cryptographic checksum.

INTEGRITY
WITH
DES

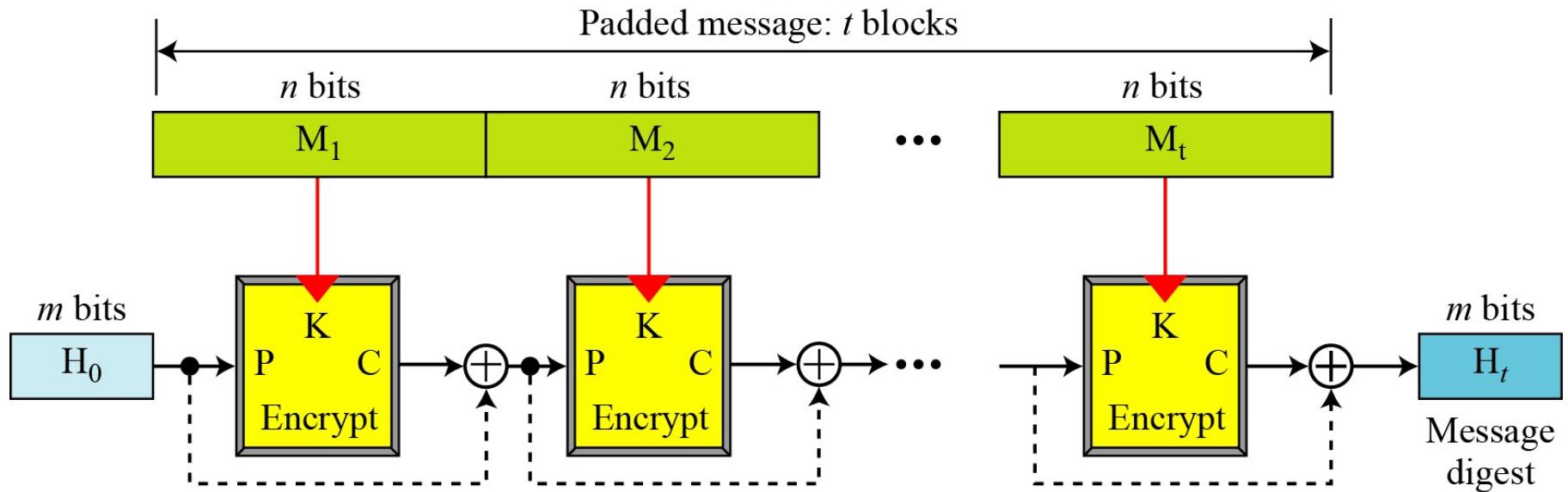
Iterated Hash with Block Ciphers

Rabin scheme



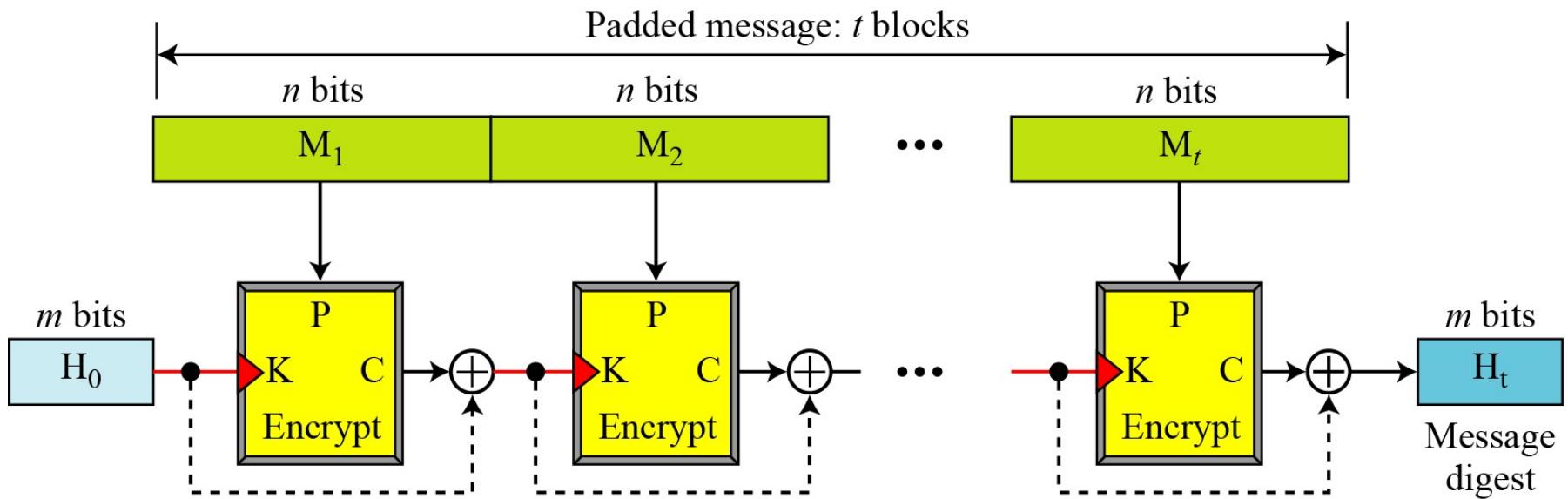
Iterated Hash with Block Ciphers

Davies-Meyer scheme



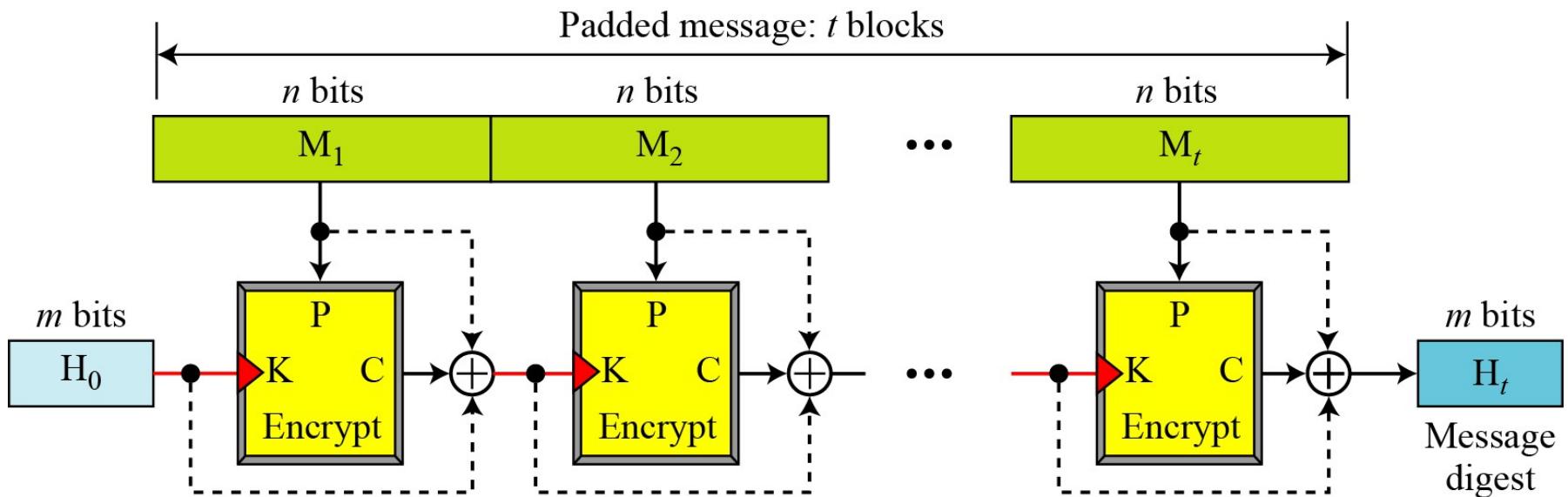
Iterated Hash with Block Ciphers

Matyas-Meyer-Oseas scheme



Iterated Hash with Block Ciphers

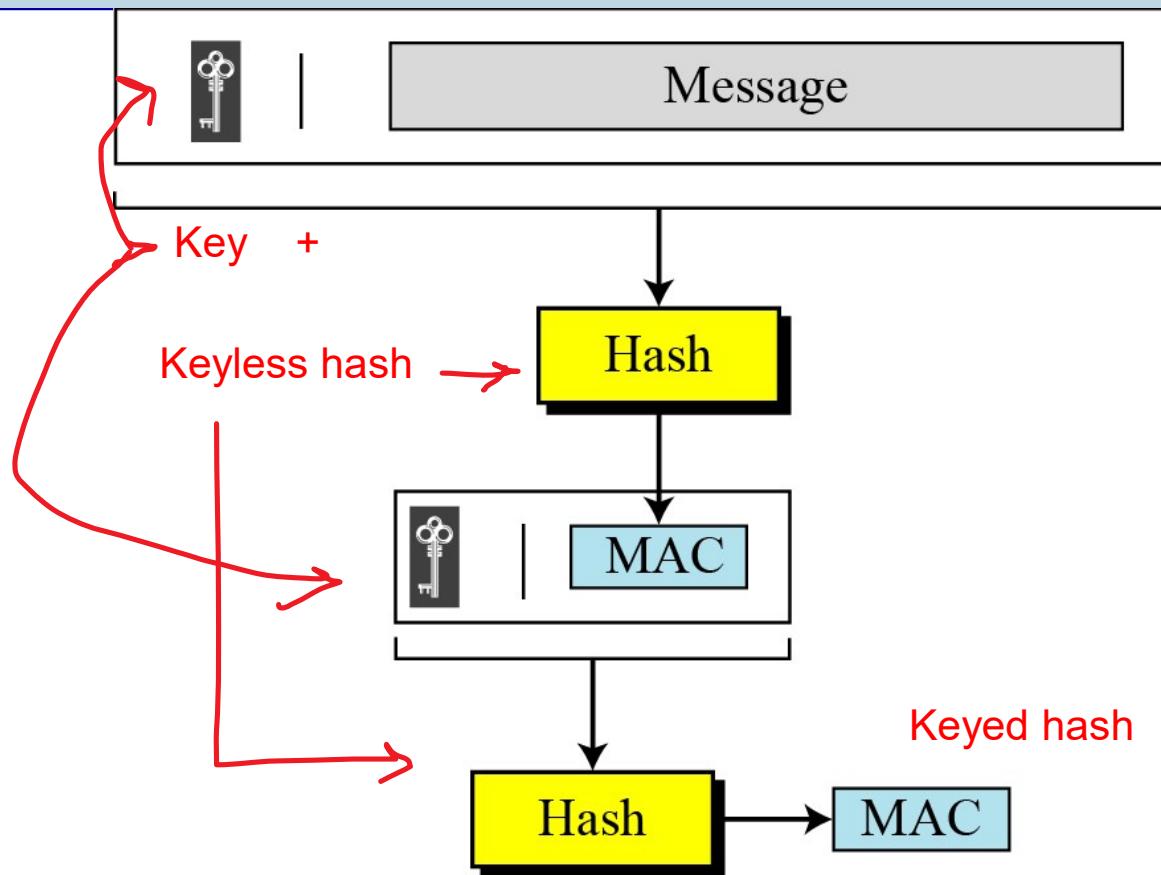
Miyaguchi-Preneel scheme



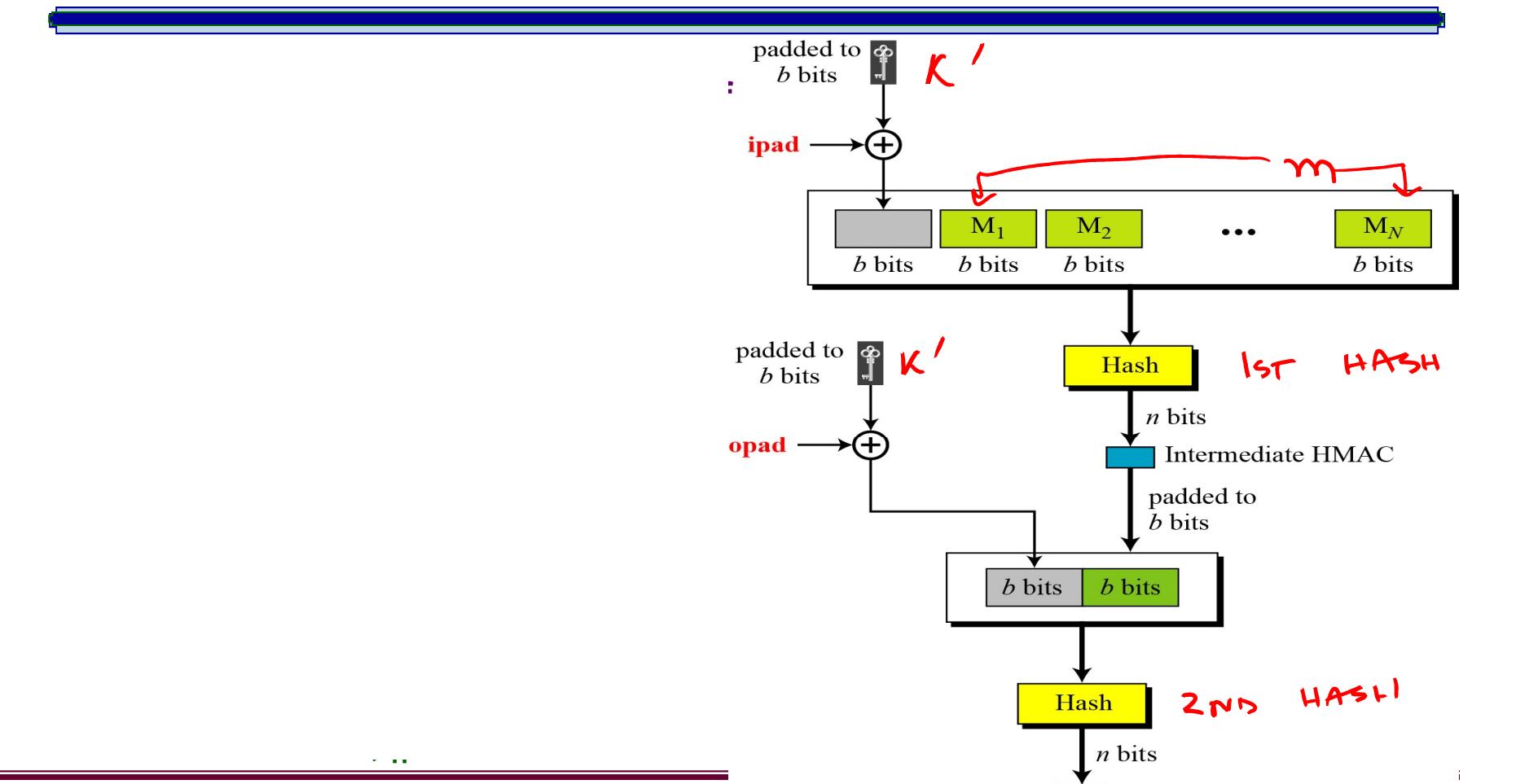
Nested Hash or HashMAC (HMAC)

- Purpose**
 - Cryptographic algorithms to produce keyed checksums had restrictions on import/export in many countries BUT keyless checksums don't <https://www.emc.com/emc-plus/rsa-labs/standards-initiatives/united-states-cryptography-export-import.htm>
 - So, idea is to use key with keyless checksums to produce keyed checksums
- General name for algorithm to produce keyed checksums (MAC) from keyless checksums (MIC)**
 - Example, if hash function is MD5, then the resulting HMAC algorithm is termed HMAC-MD5
- Strength of the HMAC primarily depends upon**
 - cryptographic strength of the underlying hash function
 - size and quality of the key
- Not affected by collision problem.**
- Uses nested checksums.**

Nested Hash Concept



HMAC



Digital Signature

- ❑ Construct that authenticates origin of message in a manner provable to a disinterested third party (“judge”)
 - Serves as evidence
- ❑ Sender cannot deny having sent message
 - Provides “non-repudiation” service
 - One could claim the cryptographic key was stolen or compromised
- ❑ Different from MAC
 - MAC do not offer non-repudiation

Example

- Classical: Alice, Bob share key k

- Alice sends $m \parallel \{m\}k$ to Bob

Is this a digital signature?

No.

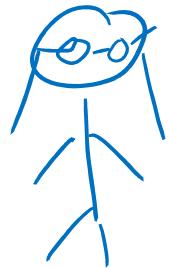
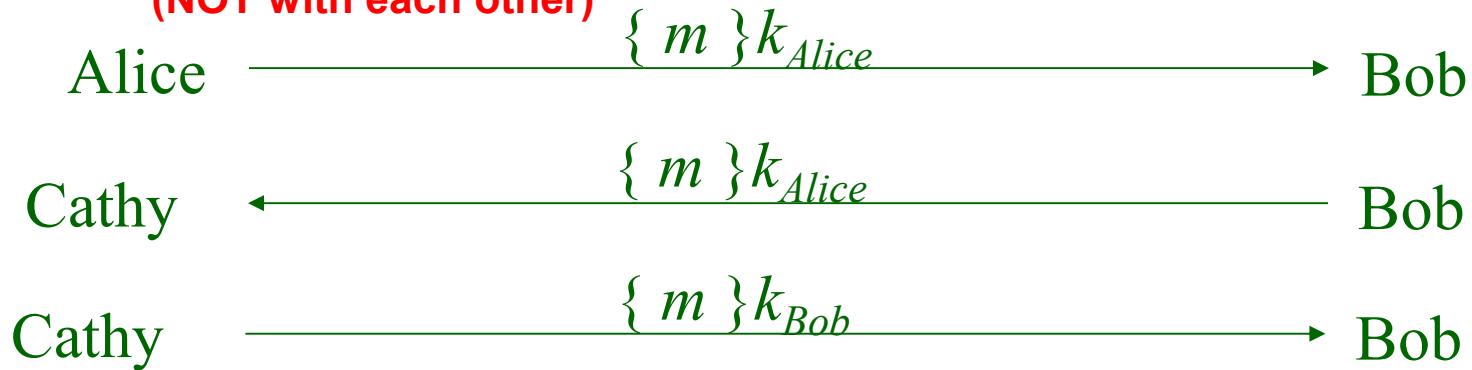
Third party cannot determine whether Alice or Bob generated message.

It is rather a MAC that supports origin integrity.

Non-Repudiation with Classical/Symmetric Cryptography

- Intervention of trusted third party is required to achieve non-repudiation with classical cryptography.

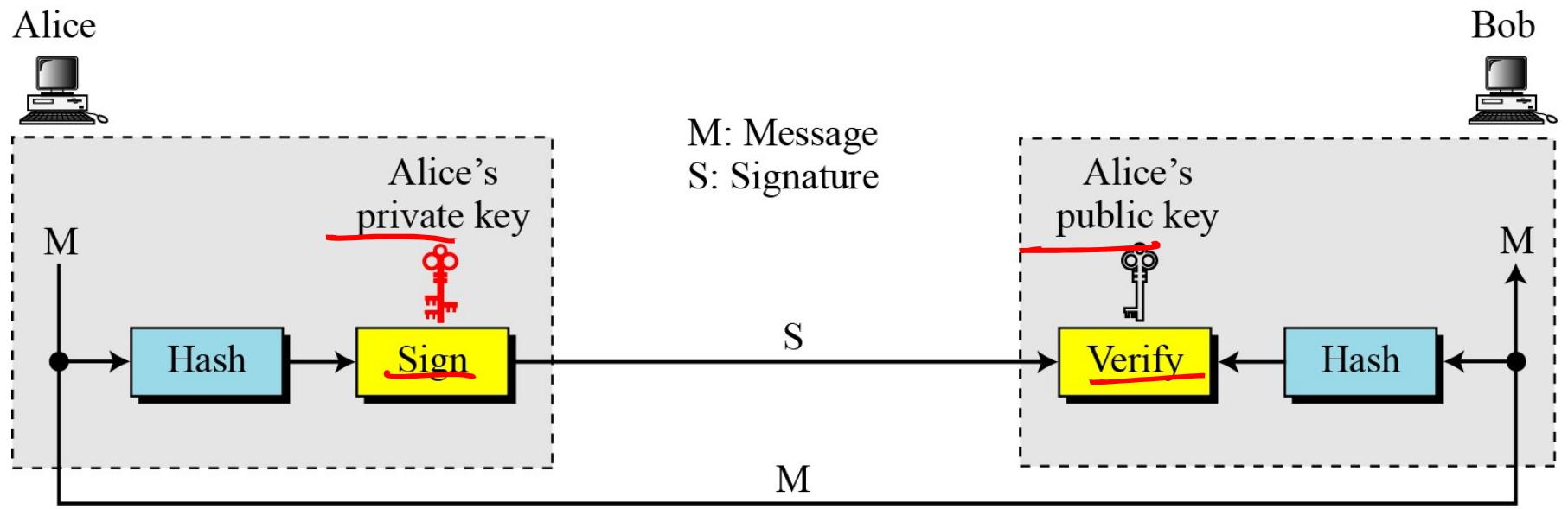
- Alice, Bob each share keys with TRUSTED 3RD PARTY Cathy (NOT with each other)



- To resolve dispute, judge gets $\{ m \} k_{Alice}$ and $\{ m \} k_{Bob}$, from Bob and Alice respectively, and has Cathy decipher them to check for forgery

Public Key Digital Signatures

- Enciphering message (or hash) with private key produces digital signature



Example

- Alice's keys are d_{Alice} , e_{Alice}
- Alice sends Bob

$$m \parallel \{ m \} d_{Alice}$$

- In case of dispute, judge computes

$$\{ \{ m \} d_{Alice} \} e_{Alice}$$

- and if it is m , Alice signed message
 - She's the only one who knows d_{Alice} !

Forgery and Precaution

- Never sign random documents
- Only signing does not help
- Should be both signed (with sender's private key) and enciphered (with receiver's public key)

- DON'T encipher and then sign
 - Sign first and then encipher

SENDER's PRIVATE

RECEIVER's PUBLIC

DOUBLE
ENCIPHERMENT

Key Differences

□ Differences between MAC and MIC

- MAC used for authentication, MIC used for integrity
- MAC requires key, MIC doesn't
 - Same algorithm on same content= Same MIC
 - Same algorithm on same content w/different key= Different MAC

□ Difference between MAC and Digital Signature

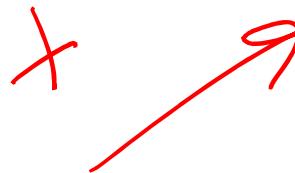
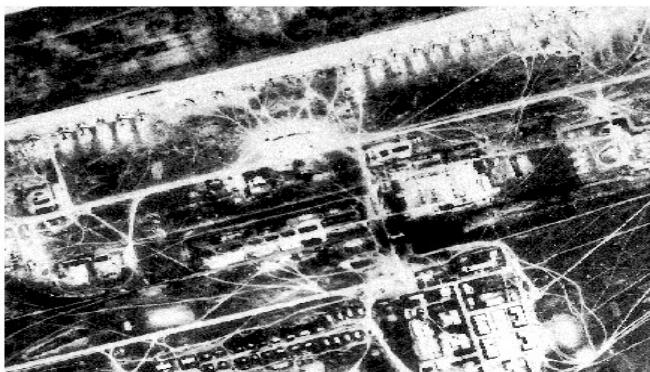
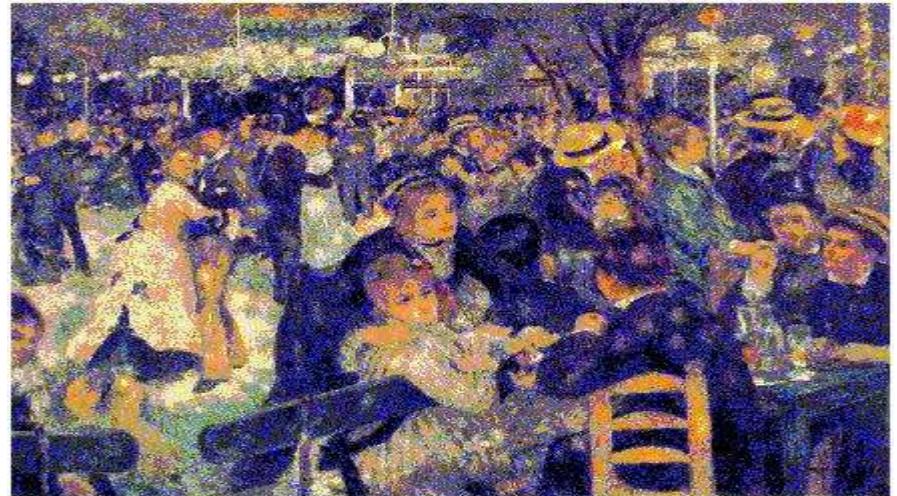
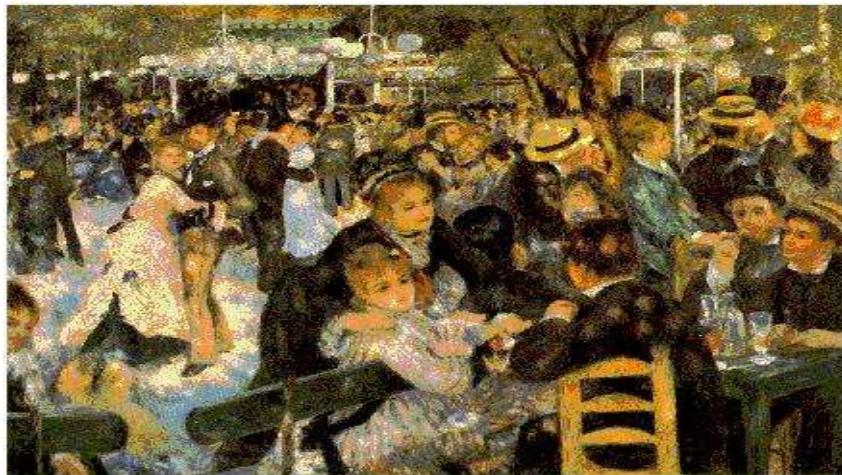
- MAC does not provide non-repudiation like digital signatures because of use of symmetric encryption

□ None provide confidentiality (if used in typical way)

Steganography

- Greek word: “covered or hidden writing”
- Hidden from unsuspecting user by embedding into message
 - Only intended recipient know of the existence of hidden message
- Differs from cryptography
 - Original message is not disguised (**Coverttext**)
 - Secret content is hidden/obscured in **Coverttext**, which becomes **Stegotext**

Example



http://www.jjtc.com/pub/tr_95_11_nfj/sec314.html

Figure 4: Long-Range Aviation Airfield⁵

Use

- **Advantage over cryptography**
 - Does not instigate curiosity/suspicion

- **Use**
 - **Benevolent use**
 - Digital watermarking, finger printing
 - Canary traps for identifying leaks

 - **Malicious use**
 - Spying, terrorism, covert communication

Steganography Techniques

- Many exists: Examples

- Manipulating noisy data
- Manipulating time/storage
- Hiding as encryption with some other encrypted data
- Chaffing and winnowing

- Lots of tools available (DataStash, MySecretFolder, Stegahide....)

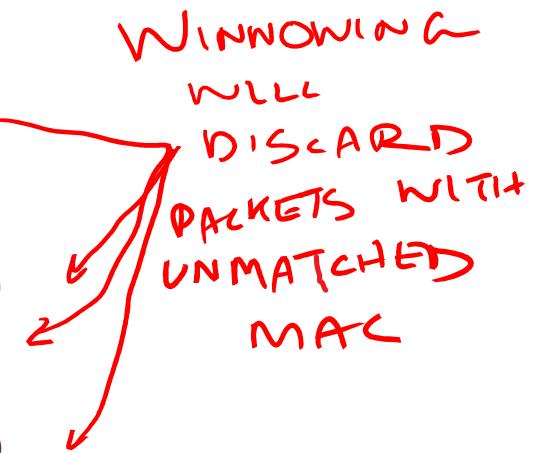
- <http://www.stegoarchive.com/>

Chaffing and winnowing

- Hiding with bogus data and then filtering bogus data
- Unique use of MAC to provide confidentiality without encryption of message
- Example:

- Original message
- (1,Hi Bob,465231) *MAC*
- (2,Meet me at,782290)
- (3,7PM,344287)
- (4,Love-Alice,312265)

- Chaffed data
 - (1,Hi Larry,532105)
 - (1,Hi Bob,465231)
 - (2,Meet me at,782290)
 - (2,I'll call you at,793122)
 - (3,6PM,891231)
 - (3,7PM,344287)
 - (4,Yours-Susan,553419)
 - (4,Love-Alice,312265)



Steganography Countermeasures

Detection of stegotext with Steganalysis

- Compare with original
- Changing format before sending
- Detection tools available (OutGuess..)

Key Points

- Two main types of cryptosystems: classical and public key
 - Classical cryptosystems encipher and decipher using the same key
 - Public key cryptosystems encipher and decipher using different keys
 - Cryptographic checksums provide a check on integrity
 - Digital signatures provide integrity of origin and content
-

Links of Interest

- <http://kathrynnneugent.com/des.html>
 - <http://www.cs.bham.ac.uk/research/projects/lems/DES/index.jsp>
 - <http://www.unixwiz.net/techtips/iguide-crypto-hashes.html>
-

Steganography

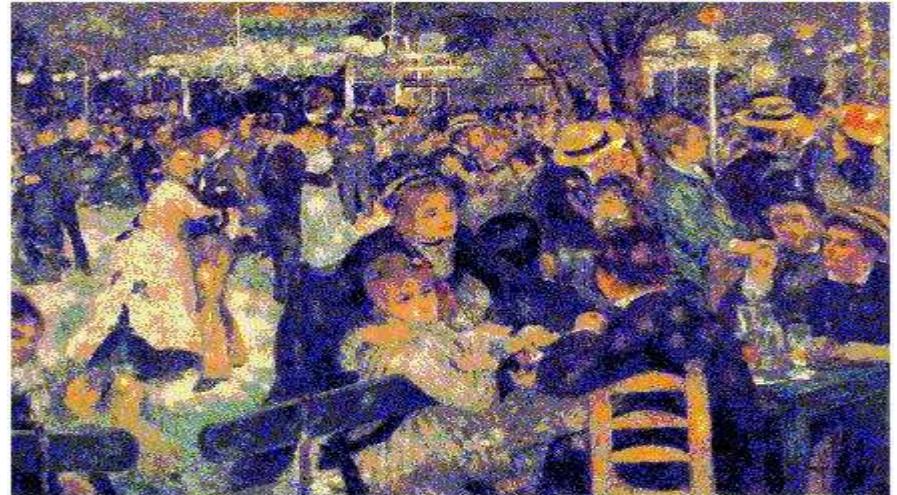
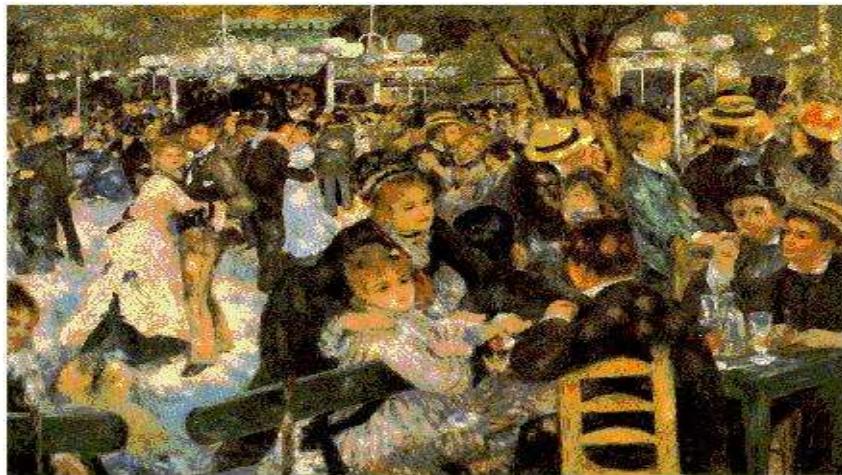
CSC 4575/5575
Information Assurance and Cryptography
Spring 2019



Steganography

- Greek word: “covered or hidden writing”
- Hidden from unsuspecting user by embedding into message
 - Only intended recipient know of the existence of hidden message
- Differs from cryptography
 - Original message is not disguised (**Coverttext**)
 - Secret content is hidden/obscured in **Coverttext**, which becomes **Stegotext**

Example



http://www.jjtc.com/pub/tr_95_11_nfj/sec314.html

Figure 4: Long-Range Aviation Airfield⁵

Use

- **Advantage over cryptography**
 - Does not instigate curiosity/suspicion

- **Use**
 - **Benevolent use**
 - Digital watermarking, finger printing
 - Canary traps for identifying leaks

 - **Malicious use**
 - Spying, terrorism, covert communication

Steganography Techniques

- Many exists: Examples

- Manipulating noisy data
- Manipulating time/storage
- Hiding as encryption with some other encrypted data
- Chaffing and winnowing

- Lots of tools available (DataStash, MySecretFolder, Stegahide....)

- <http://www.stegoarchive.com/>

Chaffing and winnowing

- ❑ Hiding with bogus data and then filtering bogus data
- ❑ Unique use of MAC to provide confidentiality without encryption of message
- ❑ Example:

- Original message
- (1,Hi Bob,465231) *MAC*
- (2,Meet me at,782290)
- (3,7PM,344287)
- (4,Love-Alice,312265)

- ❑ Chaffed data
 - (1,Hi Larry,532105)
 - (1,Hi Bob,465231)
 - (2,Meet me at,782290)
 - (2,I'll call you at,793122)
 - (3,6PM,891231)
 - (3,7PM,344287)
 - (4,Yours-Susan,553419)
 - (4,Love-Alice,312265)

WINNOWING
WILL
DISCARD
PACKETS WITH
UNMATCHED
MAC

Steganography Countermeasures

Detection of stegotext with Steganalysis

- Compare with original
- Changing format before sending
- Detection tools available (OutGuess..)

Malicious Logic

**CSC 4575/5575
Information Assurance and Security
Spring 2019**

Sources:

Introduction to Computer Security, Matt Bishop, Addison Wesley, 2003, Chapter 19

Network Security Essentials, William Stallings, 2014, Pearson

Advances in the Provision of Systems and Software Security – Thirty Years of Progress, Vaughn, 2003

Evolution of Virus and Worms

<http://lyle.smu.edu/~tchen/papers/statmethods2004.pdf>



Malicious Code/Logic

- Set of instructions that cause site security policy to be violated.

- Usually unintentionally initiated by authorized users
 - Usually assume authorized user's identity

- Example?

Malicious Code/Logic

Keylogger:

```
int main()
{
    hideConsole();
    while (true)
    {
        Sleep(20); // To make sure this program doesn't steal all
        resources.
        for (int key = 8; key <= 255; key++)
        {
            if (GetAsyncKeyState(key) == -32767)
            {
                processKey(key);
            }
        }
    }

    return 0;
}
```

Why Malicious Code Triumphs Today

- Increased
 - connectivity
 - sophistication of technology
 - availability of automated tools
- Booming underground economy
- Weakest link (people) is still weakest

Malicious Codes

- **Types**
 - Trojan Horses
 - Logic/Time Bombs
 - Viruses
 - Worms
 - Rabbits/Bacteria

Trojan Horse

- Program with an *overt* purpose (known to user) and a *covert* purpose (unknown to user)
 - Named by Dan Edwards in Anderson Report
- Remember Troy?

Propagating/Replicating Trojan Horse

- Trojan horse that makes copies of itself
 - Also called *propagating Trojan horse*
 - Early version of *animal* game (1975)

Logic Bombs

- A program that performs an action that violates the site security policy when some external event occurs
- Example:
 - Program that deletes company's payroll records when one particular record is deleted

Virus

- Program that inserts itself into one or more files and performs some action
 - *Insertion phase*
 - *Execution phase*
 - *Incubation/Dormant phase*
- Phases also known as
 - Propagation
 - Mission
 - Trigger
- May be based on conditions
 - Lehigh virus inserted itself into boot file only if boot file not infected

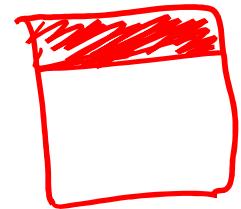
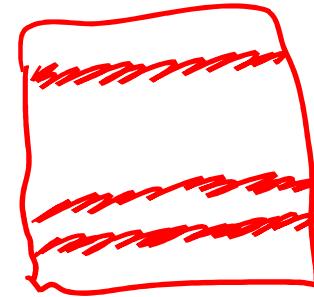
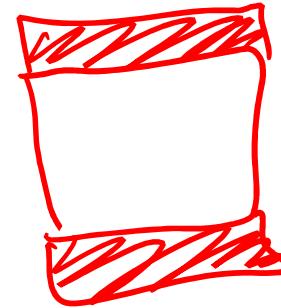
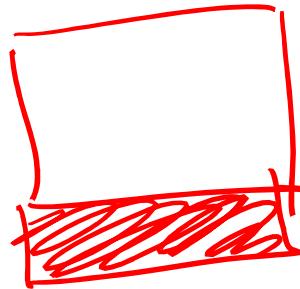
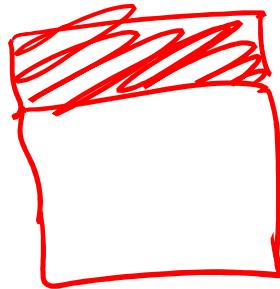
REQUIRES HOST

Pseudocode for Virus

```
beginvirus:  
    if spread-condition then begin  
        for some set of target files do begin  
            if target is not infected then begin  
                determine where to place virus instructions  
                copy instructions from beginvirus to endvirus  
                    into target  
                alter target to execute added instructions  
            end;  
        end;  
    end;  
    end;  
  
    EXECUTION  
    perform some action(s) → PAYLOAD  
    goto beginning of infected program  
endvirus:
```

Virus Insertion

PREPEND APPEND SURROUND WEAVE REPLACE



Virus: Trojan Horse Or Not?

Yes

- Overt action = infected program's original actions
- Covert action = virus' actions (infect, execute)

No

- Overt purpose = virus' actions (infect, execute)
- Covert purpose = none

Categories of Viruses

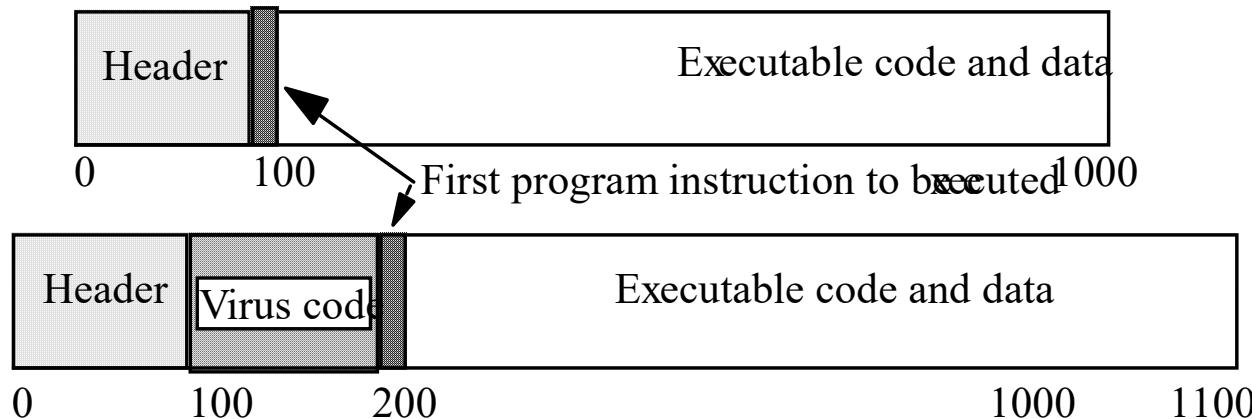
- Based on target platform
 - Boot sector
 - Executable/Parasitic
 - Multipartite (Can spread in multiple ways)
 - Based on longevity
 - TSR (Terminate and Stay Resident)
 - Based on evasion
 - Cavity
 - Stealth
 - Encrypted
 - Polymorphic
 - Metamorphic
 - Based on execution style
 - Binary
 - Macro
-

Boot Sector Infectors

- A virus that inserts itself into the boot sector of a disk
 - Executed when system first “sees” the disk
- Example
 - Brain/Pakistani virus (1986)

Welcome to the Dungeon © 1986 Basit * Amjad (pvt) Ltd.
BRAIN COMPUTER SERVICES
730 NIZAM BLOCK ALLAMA IQBAL TOWN LAHORE-PAKISTAN
PHONE: 430791,443248,280530.
Beware of this VIRUS.... Contact us for vaccination...

Executable Infectors



✓ **A virus that infects executable programs**

- Can infect .EXE or .COM files
- May prepend itself or put itself anywhere
- Example
 - Jerusalem virus (1987)

Multipartite Viruses

- ❑ A virus that can infect boot sectors or/and executables
 - There are viruses who infect only file/data
- ❑ Typically, two parts
 - One part boot sector infector
 - Other part executable infector
- ❑ Example
 - Ghostball (1989)

TSR Viruses

- TSR is “Terminate and Stay Resident”**
- A virus that stays active in memory after the application (or bootstrapping) is completed**
 - Fast and slow infectors
- Examples:**
 - Brain, Jerusalem viruses
- Counter example (Non-Memory-Resident Virus):**
 - Encroacher virus

Cavity Viruses

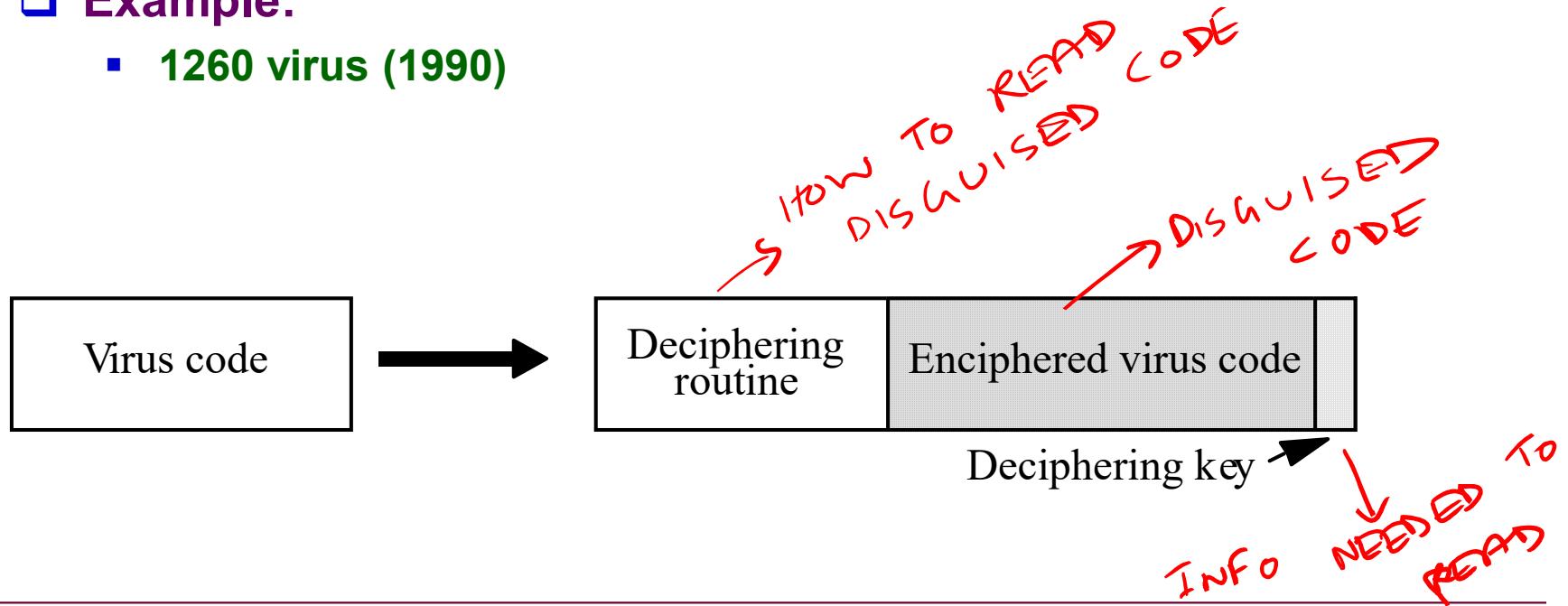
- A virus that does not change size of file
 - overwrites unused areas of executables
- Example:
 - CIH/Chernobyl virus (1998)

Stealth Viruses

- ❑ A virus that conceals infection of files
- ❑ Can intercept operating system calls to access files
- ❑ Example:
 - IDF/4096/4K virus (1989)

Encrypted Viruses

- A virus that is enciphered except for a small deciphering routine.
- Example:
 - 1260 virus (1990)



Polymorphic Viruses

- A virus that changes its form each time it inserts itself into another program
 - Polymorph at algorithm level
 - Example:
 - Dark Avenger (1988)
 - Polymorph at instruction level
 - Toolkits exist
 - Trident Polymorphic Engine
 - Mutation Engine
- mostly
DECRYPTING
ROUTINE

Instruction Level Polymorphic Example

- These are different instructions that have the same effect:
 - add 0 to operand
 - subtract 0 from operand
 - Exclusive OR 0 with operand
 - no-op
- Polymorphic virus would pick one randomly from these instructions

Metamorphic Viruses

- Completely rewrites itself after each infection
- Behavior change
 - Toolkits exist
 - Metamorphic Engine
- Example:
 - Simile (2002)

Macro Virus

- A virus composed of a sequence of instructions that are interpreted by application rather than executed directly
- Can infect
 - executables
 - Duff's shell virus
 - or data files (a.k.a., Document virus)
 - Lotus 1-2-3 spreadsheet virus
- Independent of machine architecture/platform
 - But their effects may be machine dependent
- Example
 - Melissa (1999)

Worms

- A (a stand alone) program that copies itself from one computer to another
 - Does not need host or user intervention
 - Uses network communication medium as transport
 - Email facility, remote access/execution capability, dynamic coding
 - Has phases like a virus
- Origin: Schoch and Hupp (mid '70s)
 - To serve need of distributed computations : animations, broadcast messages
 - Segment (worm) copied onto workstation
 - Segment processed data, communicated with worm's controller
 - Any activity on workstation caused segment to shut down



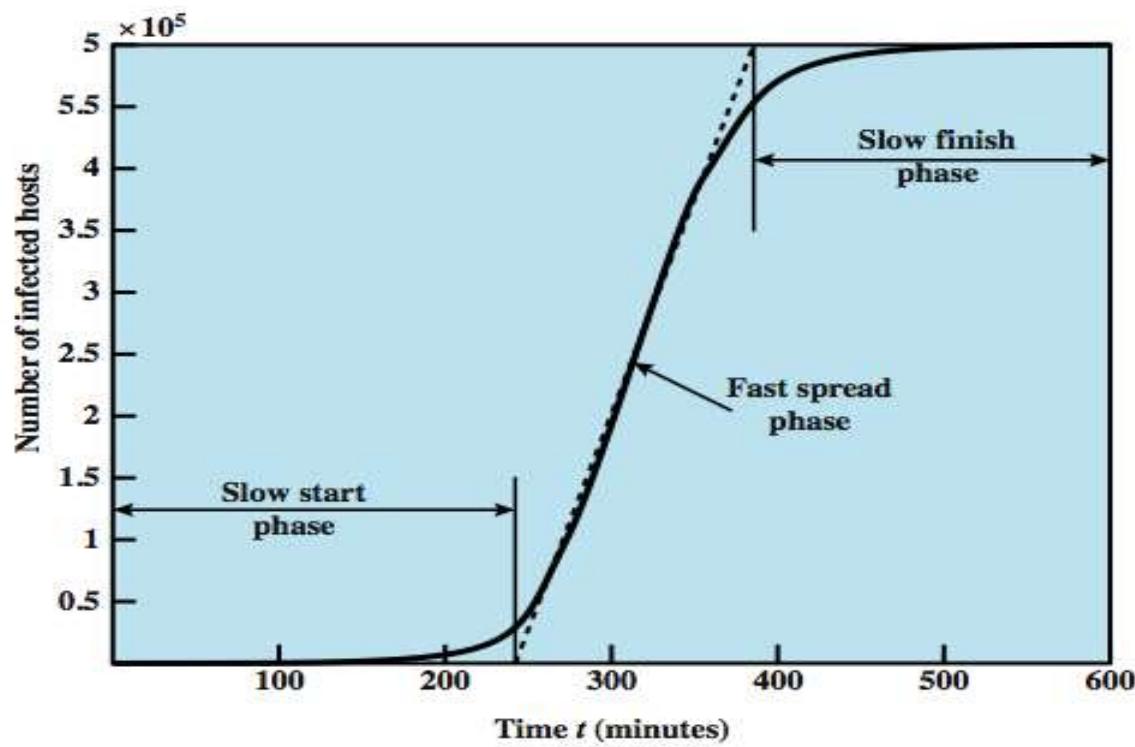
<http://pdos.csail.mit.edu/~rtm/>

Morris Worm

- Pioneer worm
 - Released by Robert Morris in 1988 through MIT machines
 - ~ 6000 unix machines infected
 - Damage was not “planned”
-



Typical Worm Propagation Rate



Modern Worm Technology

- Multi-platform
 - Multi-exploit
 - Zero-day Exploit
 - Multi-tasking
 - Transport vehicle
 - Ultrafast spreading with pre-scan
 - Dynamic
 - Polymorphic
 - Metamorphic
 - Proactive defense
-



Rabbits/Bacteria

A program that absorbs all of some class of resources

- Rabbit is usually a type of worm
- Bacteria is usually a type of virus

Example: for UNIX system, shell commands:

```
while true  
do  
    mkdir x  
    chdir x  
done
```

~~Comparison of Malicious Codes~~

- All have unauthorized intent.
- Trojan Horses
 - Hidden functionality with open legit usage
 - Do not infect
 - Some can reproduce/copy
- Logic/Time Bomb
 - Lies dormant until logic/time condition triggers
 - Do not infect
- Virus
 - Requires host to reproduce/copy
 - Infected hosts are data or program files
 - Requires user intervention to spread
- Worm
 - Independent (Do not require host or user intervention)
 - Automatically spread through network medium

Malware (or not!)

- **Rootkits**
 - Support tool for Hacker
 - Used for seizing control over OS and for concealment
- **Trap Doors/Back Doors**
 - Secret undocumented entry that bypass access control
 - Not always malicious!
 - Can be used for unauthorized unauthenticated access
- **Easter Eggs (Software Based)**
 - Secret behavior of code in response to certain stimuli
 - Kind of like logic bomb
 - Not always malicious!

Grayware

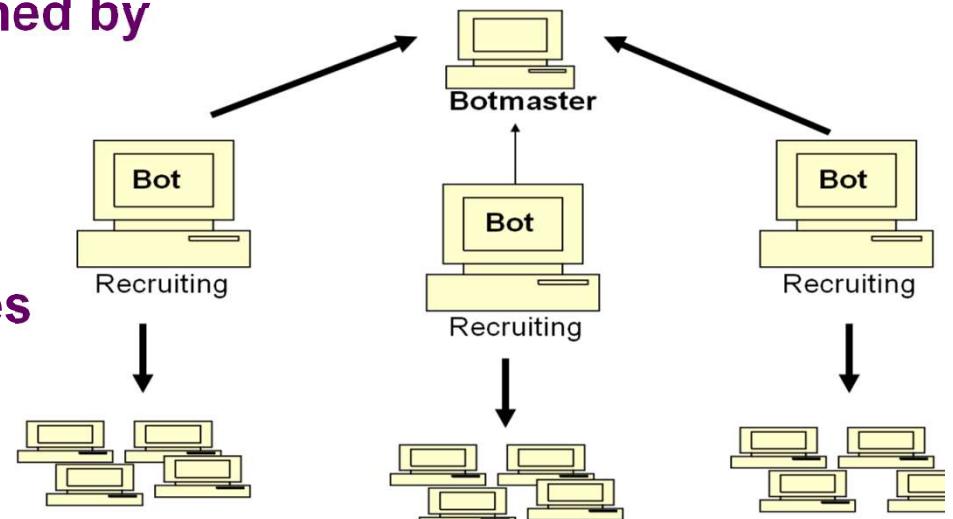
- Annoying
- Undesired
- Negatively affect performance
 - Spyware
 - Profiles users
 - » Mostly used for marketing
 - » Can be malware when collecting sensitive information
 - Adware
 - Delivers advertisement w/o user discretion
 - Mostly used for marketing

Scareware

- **Software with little or no benefit sold to naive user with unethical marketing**
 - **Exploiting people's anxiety, shock, fear**
 - **Example: Registry Cleaners**

Then there are Bots!

- A botnet is a network of infected end-hosts (bot machines) under the command of a bot master (Bot herder, mothership with human operative behind).
- 1st bot - 1999
- Bots (Zombies, Drones) claimed by
 - Exploiting vulnerabilities
 - Social Engineering
- Largest since Nov, 2012 is
Bredolab with 30 million Zombies
143 control servers



Botnet Characteristics

Lifecycle

- Recruit
- Communicate
- Execute mission

Bots commands

- Update
- Attack
 - Recruit

Uses of Botnets

- Distributed Denial-of-Service Attacks**
- Spamming**
- Sniffing Traffic**
- Keylogging**
- Spreading new malware**
- Installing Advertisement Addons and Browser Helper Objects**
 - Google AdSense abuse**
- Manipulating online polls/games**
- Mass identity theft**

<https://www.honeynet.org/book/export/html/52>



Milestones in History of Malicious Codes

- 1949 Concept of Self Reproducing Automata by John von Neumann
 - Early 1960's Malicious code used in game Darwin (Robert Morris Sr. and others from Bell Labs)
 - Late 1960's and mid-1970's 1st use of Trojan horse in penetration testing
 - Early 1970 Creeper virus (1st virus) and Reaper
 - 1974 Rabbit
 - 1975 Animal game
 - 1980 Jurgen Kraus, "Self-reproduction of programs" Ms. Thesis
 - Early 80's Xerox worm caused Denial-of Service
 - 1982 Elk Cloner (1st Apple virus)
 - Mid 80's Dr. Cohen's publication "Computer Viruses"
 - 1986/87 Brain virus (1st PC virus)
 - 1988 1st major computer security incident with wide impact (Morris Worm)
 - » 1st person (Robert Morris Jr.) to be indicted under Computer Fraud and Abuse act of 1986
 - 1989/90 1st Multipartite (Ghostball) and 1st Polymorphic virus (Chameleon)
 - 1995 1st Macro virus (Concept)
 - 2001 Metaphoramic virus (Simile)
 - 2003 Global epidemic of Internet worms (Lovesan, Sobig.f, Swen, Sober)
 - 2005 XSS virus
 - 2007 Storm worm, Zeus
 - 2009, 2010 Conficker worm, Stuxnet

~~Defense against Malicious Logic~~

- Hard to contain
 - Uses legitimate means to propagate
- Hard to detect
 - Any level of scrutiny can be defeated

Detecting Trojan Horse

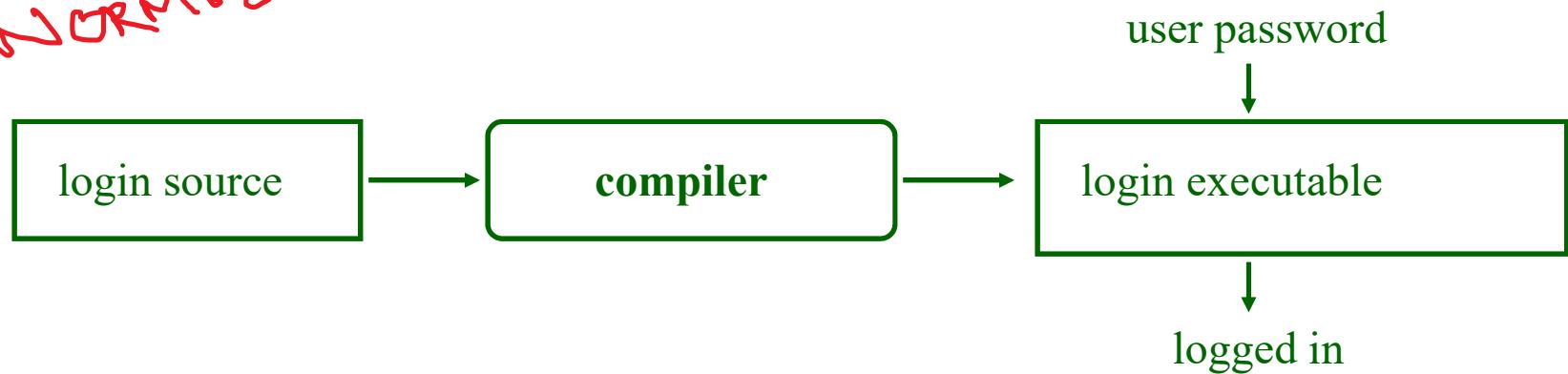
- 1976: Karger and Schell suggested modifying compiler to include Trojan horse that copied itself into specific programs including later version of the compiler
- 1980s: Thompson implements this

Thompson's Compiler

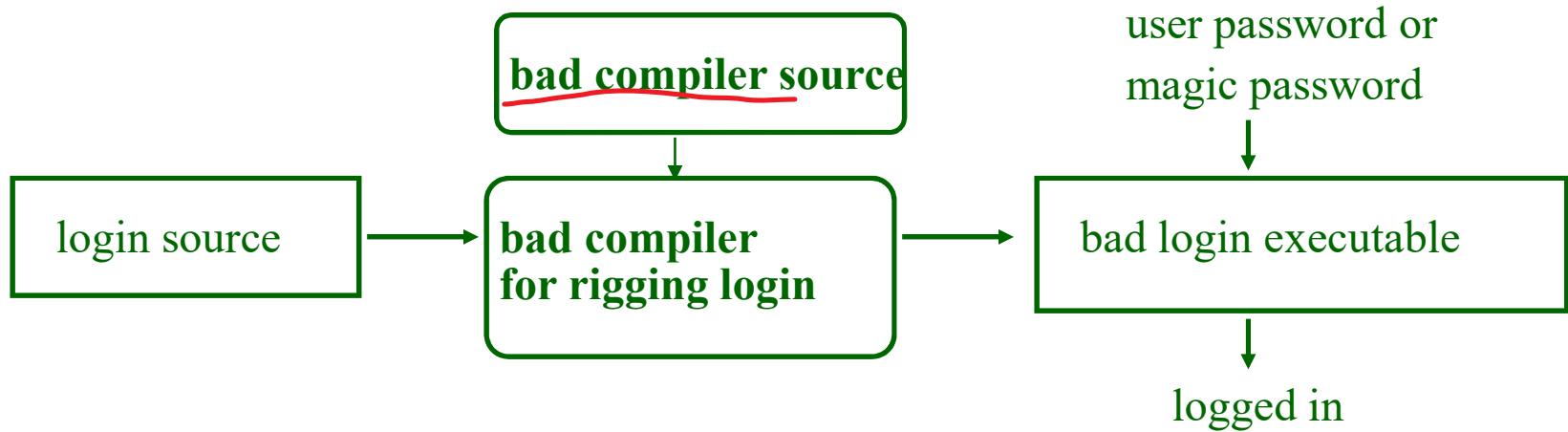
- ❑ Modify *login* source to accept the user's correct password or a fixed password (the same one for all users).
 - ❑ Code visible in login source code.
 - ❑ Instead, modify the compiler so that when it compiles *login*, *login* accepts the magic password.
 - Code visible in compiler source code.
 - ❑ Then modify the compiler again, so when it compiles a new version of itself, the extra code to do the 2nd step is automatically inserted.
 - ❑ Recompile the compiler.
 - Doctored complier in place
 - ❑ Delete the source containing the modification and put the undoctored source back.
 - Doctored complier still in place
-

The Login Program

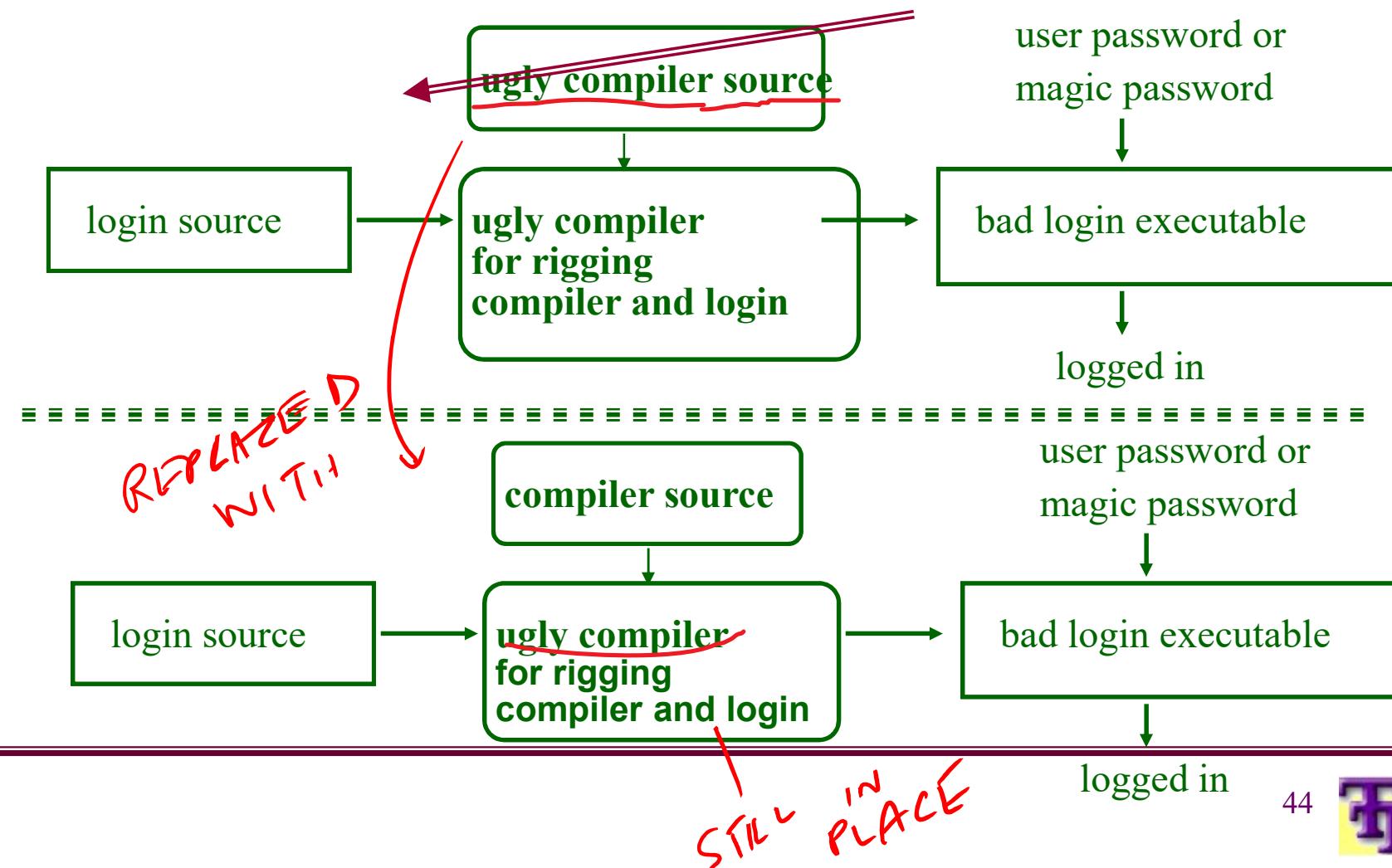
NORMAL



The BAD Compiler



The Ugly Compiler



Point is ..

No amount of source-level verification or scrutiny will protect you from using untrusted code.

Still Defend !!

- Prevent
 - Detect
 - Recover
-

Prevent

- Distinguish between data, instructions
 - Require certifying authority to convert “data” label to “instruction” label
- Limit objects accessible to processes
 - Implement least privilege principle
 - Enable reducing rights
- Inhibit sharing
 - Implement least common mechanism
- Inhibit single point of failure
 - Implement separation of privilege
 - Introduce redundancy and diversity



Prevent cont.

Filter suspicious content/activity

- Use **Firewall and/or Intrusion Prevention Systems** to detect abnormal/malicious content/activity and filter/prevent accordingly
- Use **watchdogs/guardians** to check file system events or access requests
- Use **Sandbox** to contain untrusted code
 - Tightly controlled environment with limited resource for untrusted code to run

Detect

- Detect altering of files
 - Compute and verify with manipulation detection code
 - Example: Integrity scanner Tripwire's signature block
- Detect actions beyond specifications
 - Look for known signatures
 - In code and in AntiVirus software
 - Bait files
 - Look for variations of code
 - Verify code sequences with checksum
 - Verify with proof carrying code
 - Look for anomalies in code or activity
 - Analyze characteristics to learn profile

Anti-Virus Programs

First generation

- File attribute checking, Simple pattern matching

Second generation

- File integrity checking
- Encryption checking

Third generation

- Action analysis rather than structure analysis

Fourth generation

- Containment facility with all of the above and more

Recover

- Malcode removal**
 - System Restore
 - OS reinstallation**
 - Backup**
-

~~Countermeasure Success Factors~~

- Timeliness**
 - Minimal false positives and false negatives**
 - Generality**
 - Resiliency to**
 - invasion techniques
 - evasion techniques
 - Global and local coverage**
 - Minimal operational overhead**
 - Transparency**
-

Good Virus

□ Virus that are benevolent/useful

- Can conduct virus detection, encryption and maintenance
 - “Anti-virus” virus, File Compressor virus, Disk Encryptor virus, Maintenance virus

□ Cons

- Technical
 - Lack of Control
 - Recognition Difficulty
 - Resource Occupation
- Psychological Reasons
 - Trust Problems
 - Negative Meaning
- Ethical and Legal Reasons
 - Unauthorized Data Modification
 - Copyright and Ownership Problems
 - Possible Misuse

Key Points

- A difficult problem

- How do you tell what the user did is *not* what the user intended?

- Trust underlies everything

- Policy, implementation, certification, usage

Useful Links

- http://repo.hackerzvoice.net/depot_madchat/vxdevl/vdat/epunders.htm
- <http://www.claws-and-paws.com/virus>
- <http://www.viruslist.com/en/viruses/encyclopedia>
- <http://vx.netlux.org/lib/vml00.html>
- <http://www.itsecurity.com/features/hacker-high-061008/>
- https://capitol.instructure.com/courses/510/external_tools/66

Authentication

**CSC 4575/5575
Information Assurance and Security
Spring 2019**

Sources:

Introduction to Computer Security, Matt Bishop, Addison Wesley, 2003
Thomas J. E. Schwarz, Lecture notes, Network Security, Santa Clara University
Cryptography & Network Security, B. Forouzan, McGraw Hill, 2007
How Rainbow Table Works, Kestas Kuliukas, <http://kestas.kuliukas.com/RainbowTables/>

<http://wikipedia.com>



Authentication

□ Binding of an identity to a subject/principal

- Needed for access control and accountability
 - Real time need

□ One or more of the following

- What entity knows
- What entity has
- What entity is
- Where entity is

MY PASSWORD
MY SMART CARD
MY FINGERPRINT
BRUNER 303

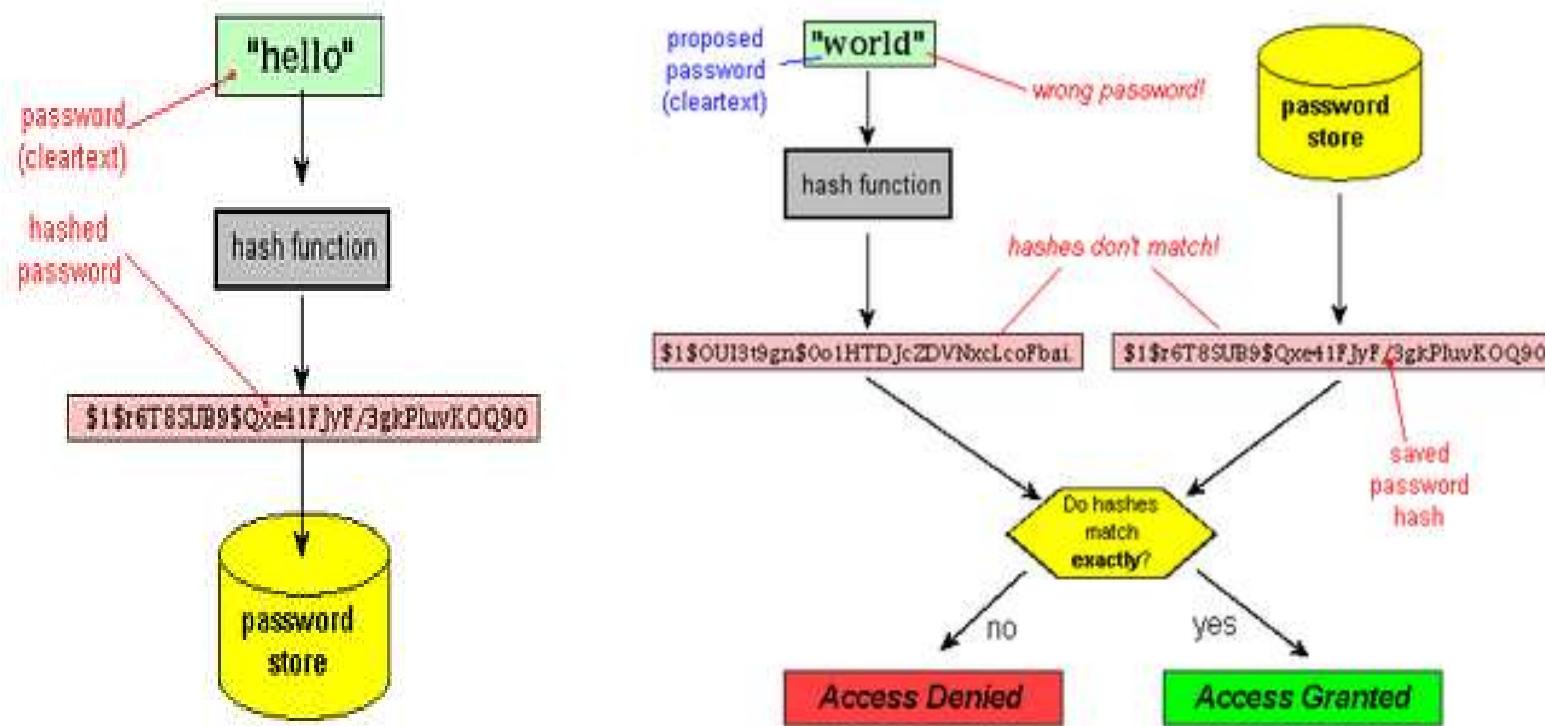
Passwords

- Sequence of characters
 - Example: 10 digits, a string of letters, etc.
- Sequence of words
 - Example: pass-phrases
- Algorithms
 - Example: challenge-response

Password Storage

- Store as clear text in file
 - If password file compromised, *all* passwords revealed
- Hide text in file
 - Encipher file
 - Need to have cipher keys in memory
 - Goes back to previous problem
 - Store one-way hashes of passwords
 - If file read, attacker must invert the hash or guess password

Passwords and Hashes



<http://www.unixwiz.net/techtips/iguide-crypto-hashes.html>

Authentication System

- **System that obtains authentication information and analyzes to verify if identity is associated with entity**
 - **(A , C , F , L , S)**
 - A information that proves identity
 - C information stored on computer and used to validate authentication information
 - F complementation function; $f : A \rightarrow C$
 - L function that verifies identity
 - S function that enables entity to create, alter information in A or C
-

Example (A , C , F , L , S) for Password system, with passwords stored as hash

- A set of strings making up passwords**
- C is the hash corresponding to A**
- F is the hash function**
- L verify identity by hash matching**
- S function to set/change password**

Unix Example

- UNIX system standard hash function
 - Hashes password into 11 char string using one of 4096 hash functions
- As authentication system:
 - $A = \{ \text{strings of 8 chars or less} \}$
 - $C = \{ 2 \text{ char hash id} \parallel 11 \text{ char hash} \}$
 - $F = \{ 4096 \text{ different versions of function based on DES} \}$
 - $L = \{ \text{login, su, ...} \}$
 - $S = \{ \text{passwd, passwd+, ...} \}$

Anatomy of Attacking

□ Goal

- Get authenticated
 - Using $a \in A$

□ Direct approach

- Get verified directly through $C \xrightarrow{\text{NEED}} \text{ACCESS TO } F, C$
 - With known $f \in F$ and C find an “ a ” such that $f(a) = c \in C$

□ Indirect approach

- Get verified through L (via C) $\xrightarrow{\text{NEED}} \text{ACCESS TO } L$
 - Get to F through $I(a)$ (....succeeds iff $f(a) = c \in C$)

Preventing Attacks

□ How to prevent this:

- Hide a, f, c
 - Example: UNIX/Linux shadow password files
 - Hides c 's
- Block access to all $l \in L$ or result of $l(a)$
 - Prevent from using $l(a)$ to guess (trial and error)
 - Example:
 - Preventing *any* logins to an account from a certain network
 - Preventing knowing results of login

Typical Password Attacks

- Try all possible passwords
 - Brute Force Attack
- Try possible passwords using short cuts
 - Optimized Brute Force
- Try probable passwords
 - Dictionary Attack

Brute Force: Finding a , given f and C

- Brute force: Knowing c , hash each possible a until you find a c that match TIME INTENSIVE
- Pre-compute: Store all possible hash or c for all possible a for future look up SPACE INTENSIVE
- Compromise
 - Rainbow table TIME MEMORY TRADE OFF

Time Memory Trade Off Analysis with Rainbow Tables

Goal

- Reduce time and memory requirements for Brute Force Attacks with pre-computed hashes

Allows faster lookup with manageable memory

Rainbow table

- Compact representation of related plaintext passwords and hashes

Password	MD5 Hash
123456	e10adc3949ba59abbe56e057f20f883e
password	5f4dcc3b5aa765d61d8327deb882cf99
12345	827ccb0eea8a706c4c34a16891f84e7b
12345678	25d55ad283aa400af464c76d713c07ad
qwerty	d8578edf8458ce06fbc5bb76a58c5ca4
123456789	25f9e794323b453885f5181f1b624d0b
1234	81dc9bdb52d04dc20036dbd8313ed055
baseball	276f8db0b86edaa7fc805516c852c889
dragon	8621ffdbc5698829397d97767ac13db3
football	37b4e2d82900d5e94b8da524fbebe33c0

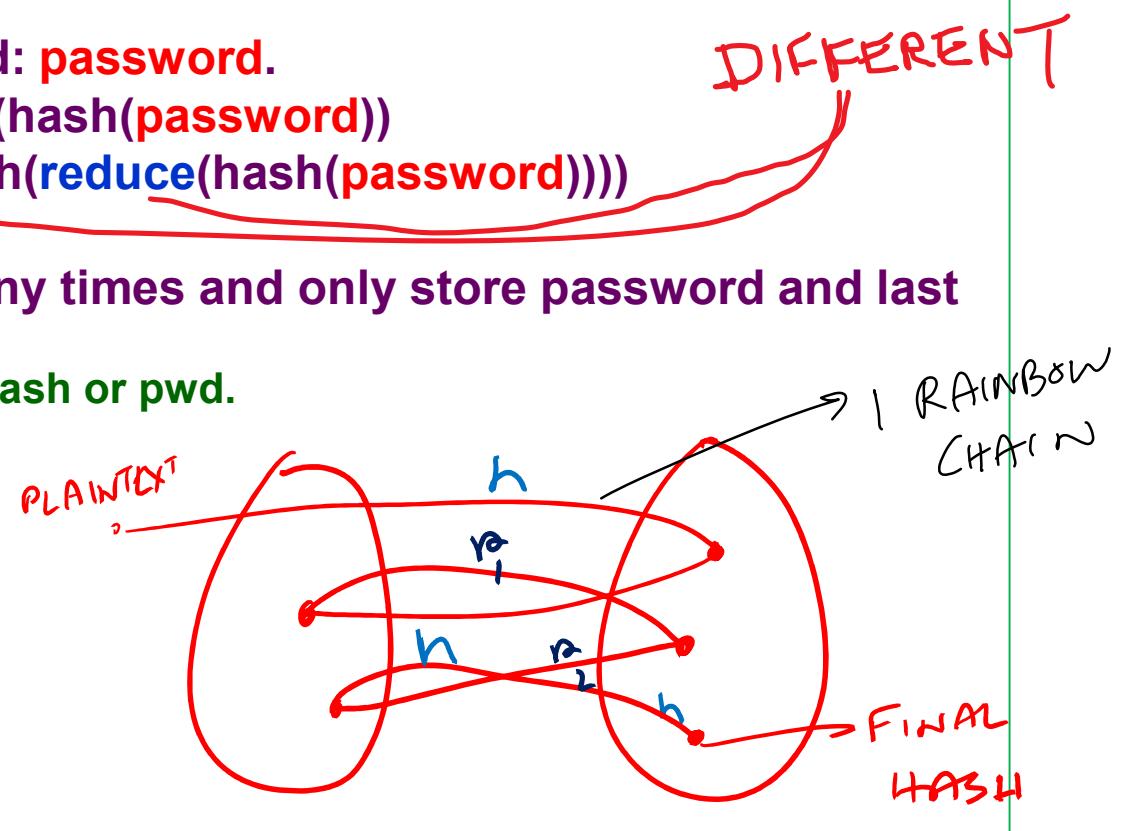
Rainbow Tables

- Stores fraction of total number of password hashes pre-computed for certain algorithms (e.g., LM, MD5)
- Table generation uses series of “reduction” functions that map hashes to different passwords (also irreversible like hashes)

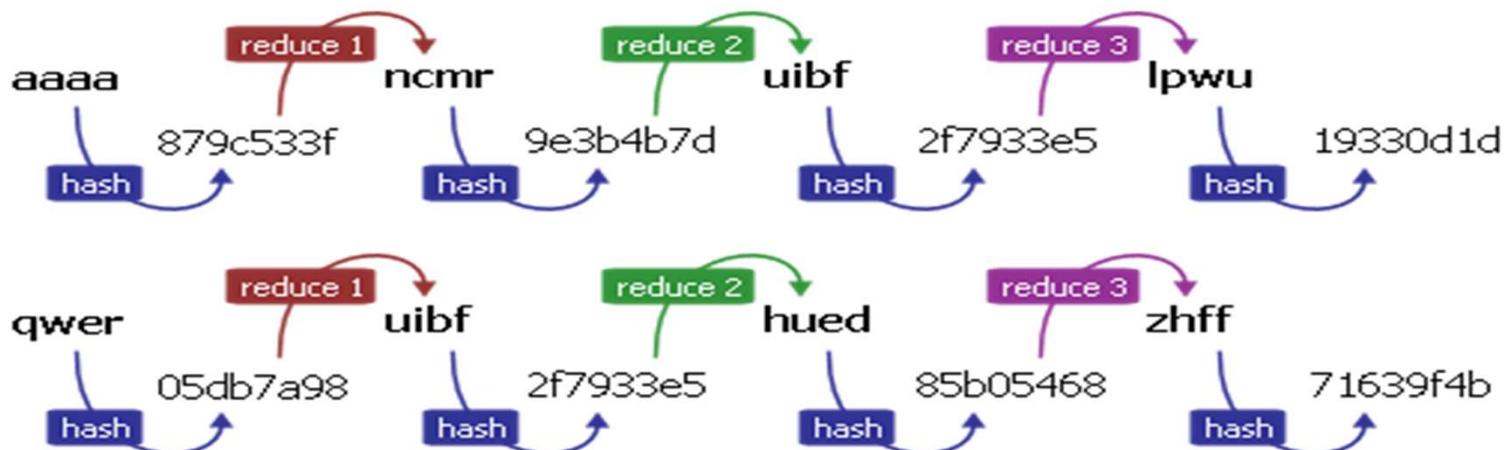


Creating Rainbow Chains

- Start with the password: **password**.
- Next element is **reduce(hash(password))**
- Next one is **reduce(hash(reduce(hash(hash(password))))))**
- ...
- Do this many many many times and only store password and last element in chain
 - Last element can be hash or pwd.



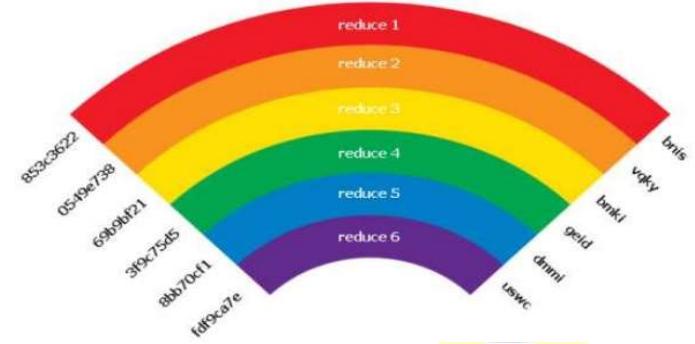
Creating Rainbow Chains



Rainbow Table

aaaa	19330d1d
qwer	71639f4b

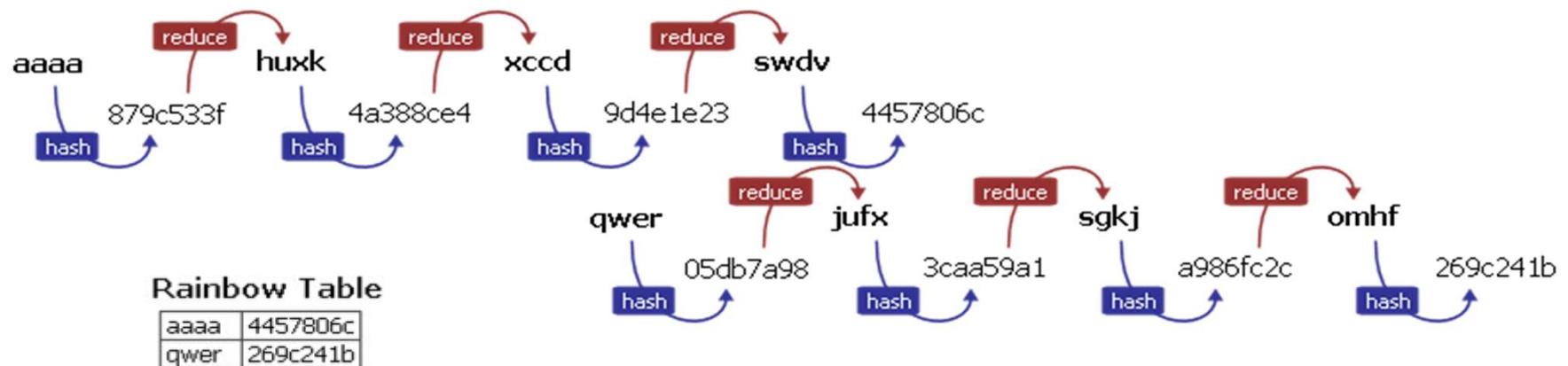
<http://www.mieuxcoder.com/2008/01/02/rainbow-tables/>



Using Rainbow Chains

- Look up table entries to match the hash h under test
 - Search hash in 2nd column,
 - If found,
 - Take the paired password P in 1st column for the matched entry
 - Apply hash-reduce iterations until the hash h is found
 - The immediate password P' that directly generated hash h is the answer.
 - If not found, calculate hash(reduce (h))
 - Repeat (until a match is found).

Using Rainbow Chains Example

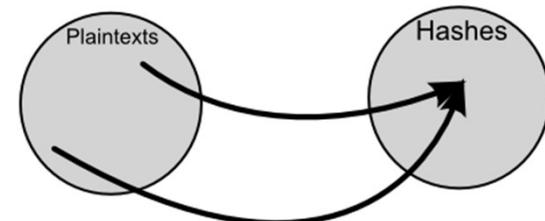


- If hash is: **269c241b**, entry found in table
 - (a) Start with **qwer** and then the sought password is recomputed and found to be **omhf**
- If hash is: **4a388ce4**, entry NOT found in table
 - Keep reducing and hashing until one entry is found (**4457806c**) in table
 - Repeat as (a) until password is recomputed and found to be **huxk**
- Not guaranteed success (probabilistic)

Rainbow Table Defense

❑ Collision

- In this case, having collisions is a good thing because rainbow chains merge with ambiguous results



<http://kestas.kuliukas.com/RainbowTables/>

❑ Salting

- Makes pre-computation useless

Dictionary Attacks

Off-line (Type 1):

- Attacker knows f and c 's, and repeatedly try different guesses $g \in A$ until any matches a c
- Examples: *crack, john-the-ripper*
 - Usually does not have a specific target
 - Needs to have access to f and c
 - Can cause potentially greater damage

On-line (Type 2):

- Attacker has access to functions in L and try guesses g until some $l(g)$ succeeds
- Examples: trying to log in by guessing a password
 - Usually specific target
 - Needs to have access to $l(g)$
 - Damage limited (unless root)

Password Guessing Steps

- No password
 - User ID
 - User name
 - User names' variations
 - User information
 - Common word list
 - Common patterns
 - Short college dictionary
 - Complete English word list
 - Common non-English language dictionaries
 - Short college dictionary with capitalizations
 - Complete English with capitalizations and substitutions
 - Common non-English language dictionaries with capitalizations and substitutions
 - Brute-force
-

Password guessing is possible because ...

- People choose easy to guess passwords**
 - Based on account names, user names, computer names, place names
 - Dictionary words
 - Too short, digits only, letters only
 - License plates, acronyms, social security numbers
 - Personal characteristics (children name, parents names, pet names, nicknames, job characteristics, etc.)

Countering password guessing

Anderson's formula:

- P probability of guessing a password in specified period of time
- G number of guesses tested in 1 time unit
- T number of time units
- N number of possible passwords
- Then $P \geq TG/N$

$$P \downarrow \quad \text{and} \uparrow$$

Example

□ Goal

- Passwords drawn from a 96-char alphabet
- Can test 10^4 guesses per second
- Probability of a success to be 0.5 over a 365 day period
- What is minimum password length?

□ Solution

- $N \geq TG/P = (365 \times 24 \times 60 \times 60) \times 10^4 / 0.5 = 6.31 \times 10^{11}$
- Choose s such that $\sum_{j=0}^s 96^j \geq N$
- So $s \geq 6$, meaning passwords must be at least 6 chars long

$$A^S=N$$

A: Alphabet size, S= Password size

✓ Protection of Type (1) Dictionary Attacks: Salting

- Goal: mitigate dictionary attacks.
- Types:
 - randomly select a hash function out of multiple choices
 - Concatenate random data with password as input to the hash algorithm
- Increases work for attacker
- Example:
 - Vanilla UNIX method
 - Perturb hash function in one of 4096 ways

$$C = \{ 2 \text{ char hash id} \parallel 11 \text{ char hash} \}$$
$$F = \{ 4096 \text{ different versions of modified DES} \}$$

Protection of Type (2) Dictionary Attacks

- Cannot prevent
 - Otherwise, legitimate users cannot log in
- Deter
 - Backoff
 - Delay next attempt
 - Disconnection
 - Disconnect after threshold
 - Disabling
 - Disable after threshold
 - Extra caution with administrative accounts!
- Deflect
 - Jailing
 - Allow with restriction

Other Defenses

- Against password cracking
 - Proactive password checking

- Against password replay
 - Password aging
 - Challenge response

Proactive Password Checking

- Analyze proposed password for “goodness”
- Criteria
 - Always invoked
 - Easy to set up and integrate into password selection system
 - Detect, reject bad passwords for an appropriate definition of “bad”
 - Discriminate per-user, per-site basis
 - Conduct pattern matching on words
 - Execute subprograms and use results
 - For example, Spell checker

Example: *passwd+*

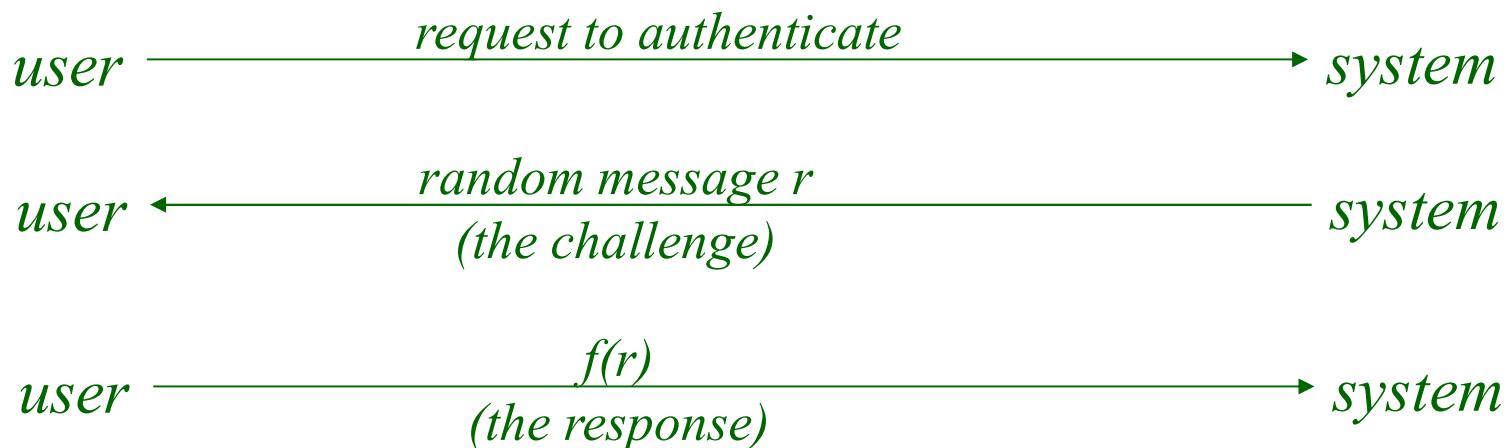
- Provides mechanism for proactive checking
 - `test length("$p") < 6`
 - If password under 6 characters, reject it
 - `test infile("/usr/dict/words", "$p")`
 - If password in file /usr/dict/words, reject it
 - `test !inprog("spell", "$p", "$p")`
 - If password not in the output from program spell, given the password as input, reject it (because it's a properly spelled word)

~~Password Aging~~

- A requirement to force users to change passwords after some time has expired
 - How often should change occur?
 - Depends on policy, practice and mechanism used
 - How do you force users not to re-use passwords?
 - Record previous passwords
 - Give users time to think of good passwords
 - Warn them of expiration days in advance

Challenge-Response

- User, system share a secret function f (f maybe a secret function or a known function with unknown parameters, such as a cryptographic key)



- If system's computation of $f(r)$ matches with user sent $f(r)$, authentication success

Challenge-Response

- Pass algorithms
 - One time passwords
 - Hardware supported
-

Pass Algorithms

- Challenge-response with the function f itself a secret
 - Example:
 - Challenge is a random string of characters such as “abcdefg”, “ageksido”
 - Response is some function of that string such as “bdf”, “gkio”

One-Time Passwords

- **Ultimate password aging**
- **Password that can be used exactly once**
 - After use, it is immediately invalidated
- **Can also be used in challenge-response manner**
 - Challenge is number of authentications identifying the authentication session; response is the password for that particular number
- **Problems**
 - Generation of good random passwords
 - Synchronization of user, system

One-time Password Scheme: S/Key

- Based on idea of Lamport
 - Used in Unix like operating systems
 - h one-way hash function (MD5 or SHA-1, for example)
 - User chooses initial seed k
 - System calculates:
$$h(k) = k_1, h(k_1) = k_2, \dots, h(k_{n-1}) = k_n$$
 ℓ_1 ℓ_{n-1} ℓ_n
 - Passwords are used in reverse order:
$$P_n, P_{n-1}, \dots, P_1$$
 - System only store last password (p_n)
 - User is given the list of passwords in reverse order and starts with last but one (p_{n-1})
-

S/Key Protocol

KEY GENERATION

P_1

$$\downarrow h(P_1)$$

P_2

$$\downarrow h(P_2)$$

P_3

\vdots

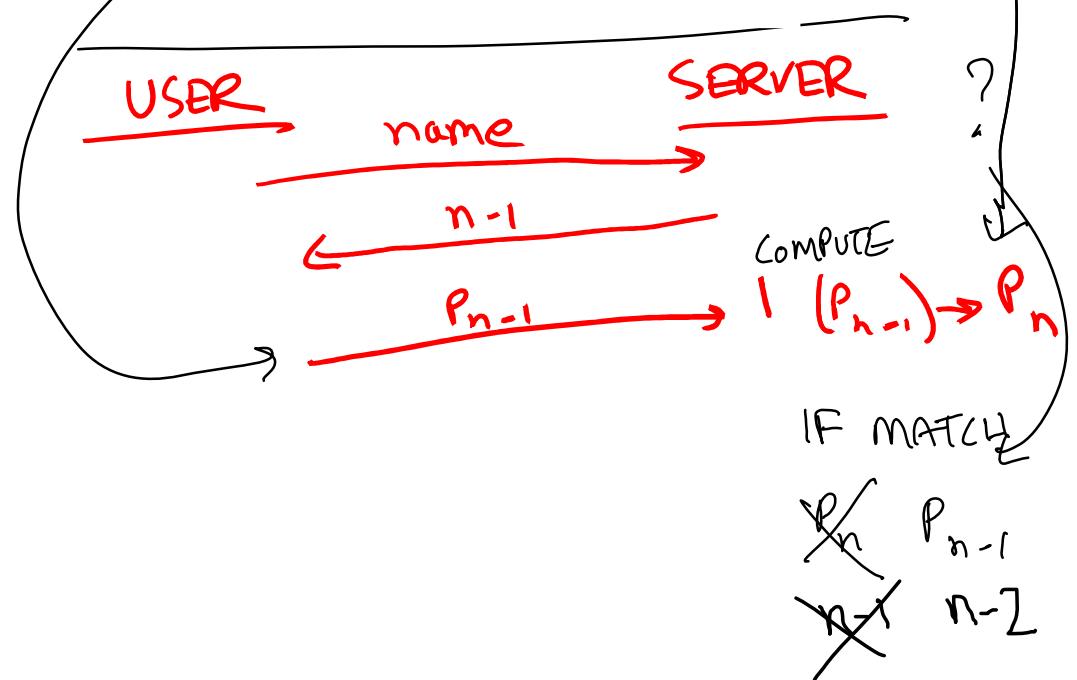
\vdots

$$P_{n-1} \downarrow h(P_{n-1})$$

P_n

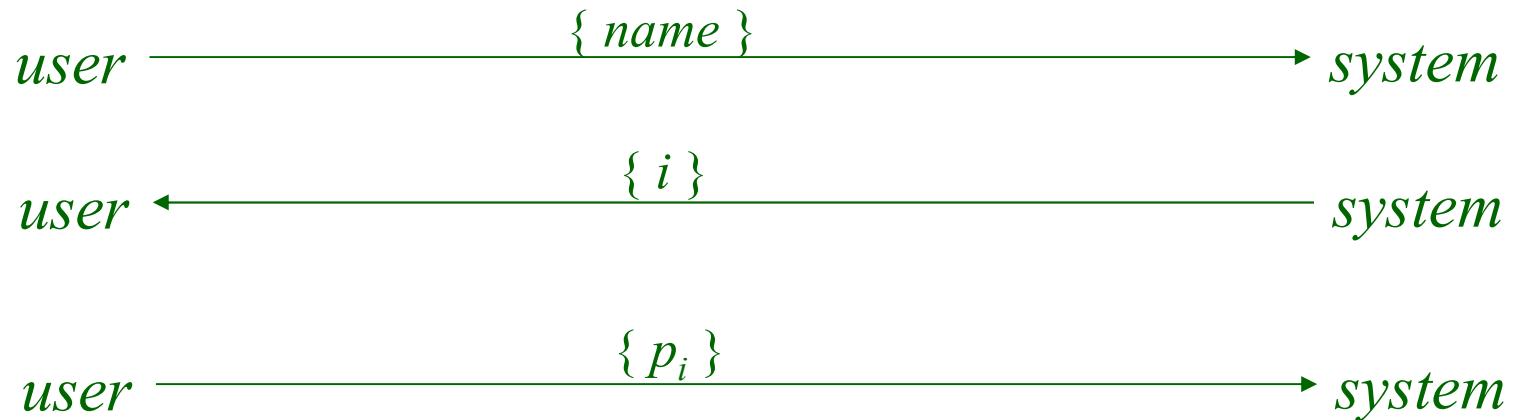
USER HAS
 $P_{n-1}, P_{n-2}, \dots, P_1$

SERVER KNOWS
 (n, P_n)



S/Key Protocol – Another View

System stores maximum number of authentications n , number of next authentication i , last correctly supplied password p_{i-1} .



System computes $h(p_i) = p_{i-1}$. If match with what is stored, system replaces p_{i-1} with p_i and updates i .

Hardware Supported Authentication

- What entity has

- Token-based

- User owned device used to compute response to challenge sent by computer

- Computer knows what response to expect

- May encipher or hash challenge with user supplied PIN

- Temporally-based

- Every minute (or so) different number shown in user owned device

- Computer knows what number to expect when

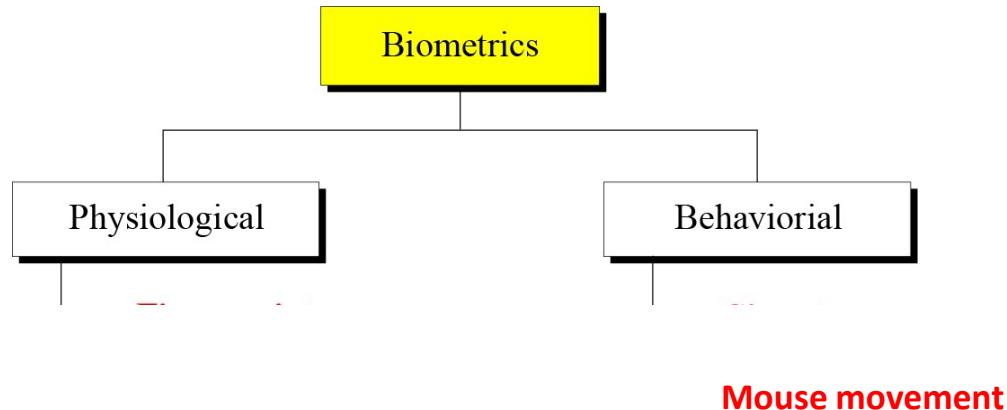
- User enters number and fixed password

- Example: RSA SecureID

Biometrics

SOMETHIN'
INHERENT

- What entity is
- Automated measurement of biological, behavioral features that identify a person by comparison



Characteristics

- **Fingerprints: optical or electrical techniques**
 - Maps fingerprint into a graph, then compares with database
- **Voices: speaker verification or recognition**
 - uses statistical/AI techniques to test hypothesis that speaker is who is claimed
 - checks content of answers
- **Eyes**
 - Patterns in irises unique
- **Faces**
 - Image, or specific characteristics like distance from nose to chin
- **Keystroke and mouse dynamics**
 - Keystroke intervals, pressure, duration of stroke, where key is struck, mouse movements

Cautions

- These can be fooled!
 - Assumes biometric device accurate *in the environment it is being used in!*
 - Transmission of data to validator is tamperproof, correct
- These can be erroneous!

Location

- ❑ Where entity is
- ❑ Validate identity by verifying location
 - Where you are and what you have
 - Requires special-purpose hardware to locate user
 - GPS (Global Positioning System) provide location signature of user (signature includes time and place)
 - Computer gets location signature for user from GPS
 - User uses LSS (Location Signature Sensor) to get location signature for his/herself
 - Send to computer
 - If match, validated
 - Problem if users' location signature generating device stolen/compromised
 - unless policy in place to check further

Multiple Methods

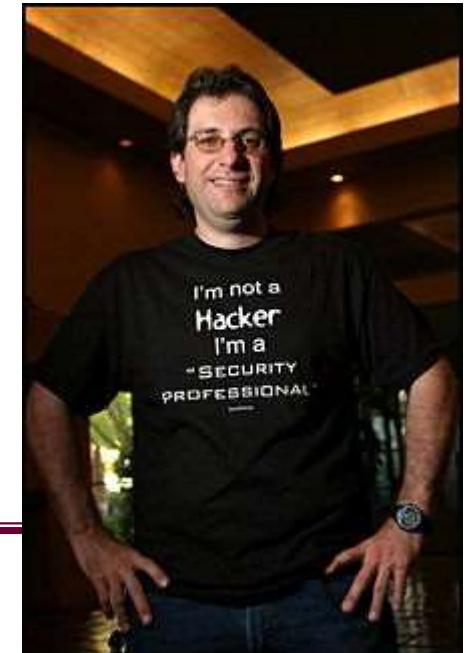
- Authentication methods can be combined
 - Something you know + something you possess
 - Smart cards in combinations with passwords
 - Something you are + something you know
 - Biometrics in combinations with passwords
 - Something you possess + somewhere you are
 - LSS in combinations with location

Good password practice

- Use “strong” password**
 - Long passwords (length at least 8)
 - At least 1 digit, 1 letter, 1 punctuation symbol, 1 control character
 - Be creative
 - “LIMm*2^Ap”
 - Names of members of 2 families
 - “OoHeO/FSK”
 - Second letter of each word of length 4 or more in third line of third verse of Star-Spangled Banner, followed by “/”, followed by author’s initials
- Change password**
- Don’t reuse password for different systems**
- Don’t use “remember password”**
- Don’t write them down**
- Don’t tell anyone**
 - Social Engineering

Social Engineering

- Art of deception
- “Using manipulation, influence and deception to get a person, a trusted insider within an organization, to comply with a request, and the request is usually to release information or to perform some sort of action item that benefits that attacker”.
 - [Kevin Mitnick](#)
 - » Notorious ex-hacker, Famous security consultant
 - » [Master of Social Engineering](#)



Techniques (Online & Offline)

- Dumpster Diving**
 - Using carelessness in disposing of sensitive information
 - Pretexting**
 - Using invented scenario (pre-text) to persuade a target to release sensitive information or perform an action that violates policy
 - Phishing**
 - Email appearing to come from a legitimate business and responding to it may result in disclosing sensitive information through a fraudulent web site
 - Vishing (VOIP), Smishing (SMS), Spear Phishing, Whaling
 - Gimmes**
 - Exploiting curiosity/carelessness to deliver malware
 - Quid pro Quo**
 - Offering technical help/goods for information
 - Shoulder surfing**
 - Smoking zone**
 - Piggybacking**
 - Reverse social engineering**
 - Sabotage + Advertising + Assisting
-

Dealing with Social Engineering

- Easier for attackers to use than technical means

- Very hard to prevent as it exploits human characteristics as
 - **carelessness, helpfulness, helplessness, comfort zone, fear, curiosity, greed, etc.**

carelessness, helpfulness,
helplessness, comfort zone, fear,
curiosity, greed, etc.

Techniques (Online & Offline)

- Dumpster Diving**
 - Using carelessness in disposing of sensitive information
 - Pretexting**
 - Using invented scenario (pre-text) to persuade a target to release sensitive information or perform an action that violates policy
 - Phishing**
 - Email appearing to come from a legitimate business and responding to it may result in disclosing sensitive information through a fraudulent web site
 - Vishing (VOIP), Smishing (SMS), Spear Phishing, Whaling
 - Gimmes**
 - Exploiting curiosity/carelessness to deliver malware
 - Quid pro Quo**
 - Offering technical help/goods for information
 - Shoulder surfing**
 - Smoking zone**
 - Piggybacking**
 - Reverse social engineering**
 - Sabotage + Advertising + Assisting
-

Dealing with Social Engineering

- **No single solution**
 - Needs awareness/education
 - Needs training
 - Needs policy placement and enforcement
 - Risk management and contingency plans
 - Legal actions

- **“You could spend a fortune purchasing technology and services...and your network infrastructure could still remain vulnerable to old-fashioned manipulation.” Mitnick**

Key Management

CSC 4575/5575
Information Assurance and Security
Spring 2019

Sources:

Introduction to Computer Security, Matt Bishop, Addison Wesley, 2003
Security in Computing, Pfleeger and Pfleeger, Prentice Hall, 2003
Cryptography & Network Security, B. Forouzan, McGraw Hill, 2007

Key Management

- Distribution of cryptographic keys
 - Mechanisms to bind an identity to a key
 - Generation, maintenance and revocation of keys
-

Notation

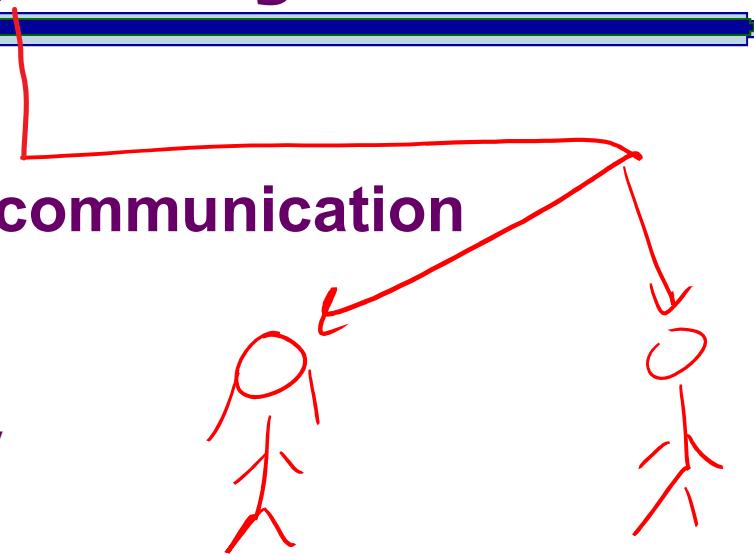
- $X \rightarrow Y : \{ Z \parallel W \} k_{X,Y}$
 - X sends Y the message produced by concatenating Z and W enciphered by key $k_{X,Y}$, which is shared by users X and Y
- $A \rightarrow T : \{ Z \} k_A \parallel \{ W \} k_{A,T}$
 - A sends T a message consisting of the concatenation of Z enciphered using k_A , A 's key, and W enciphered using $k_{A,T}$, the key shared by A and T

Types of Keys

- Interchange key
- Session key

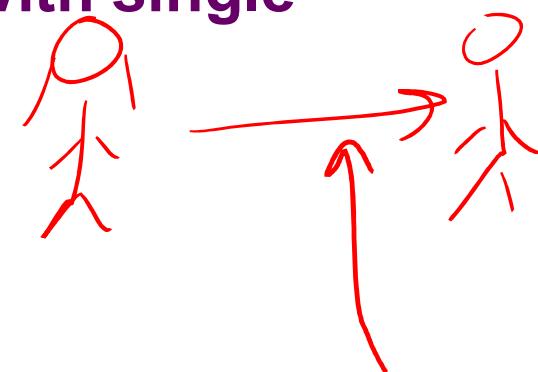
Interchange Key

- Vouches for a user/principal
- Associated with principal(s) in a communication
 - Principal is user communicating
 - In case of classical cryptography
 - Interchange key is shared secret key
 - In case of public key cryptography
 - Interchange key is public, private pair



Session Key

- Associated with a single communication
- Discarded afterwards
- Limits amount of traffic enciphered with single key
 - Used per communication session
- Helps with
 - Forward search/precomputation attack
 - Sniffing attack
 - Replay attack



Need for Key Exchange Protocol

- Goal: Alice, Bob need to get shared session key to communicate messages
 - Session key cannot be sent in clear
 - Session key can be sent enciphered with interchange key

(x_s)

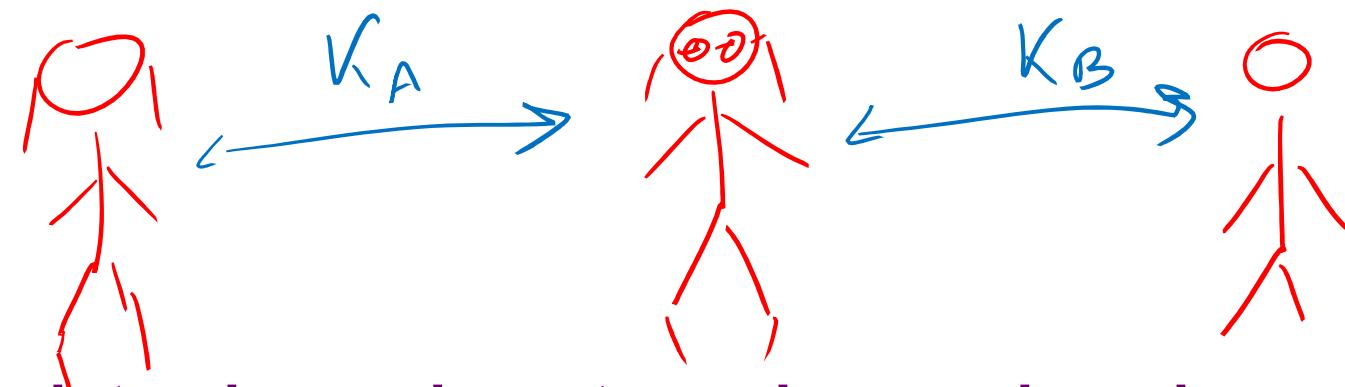
$\{k_s\}_{k_i}$

- Classical and public key exchange protocols are different

Classical (Session) Key Exchange

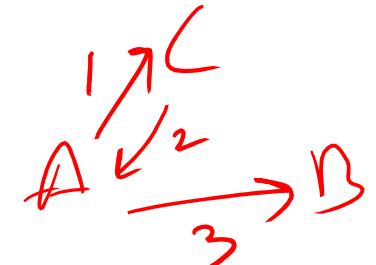
□ Assume trusted third party, Cathy

- Alice and Cathy share secret key k_A
- Bob and Cathy share secret key k_B



□ Use interchange keys to exchange shared session key k_s

Simple Protocol



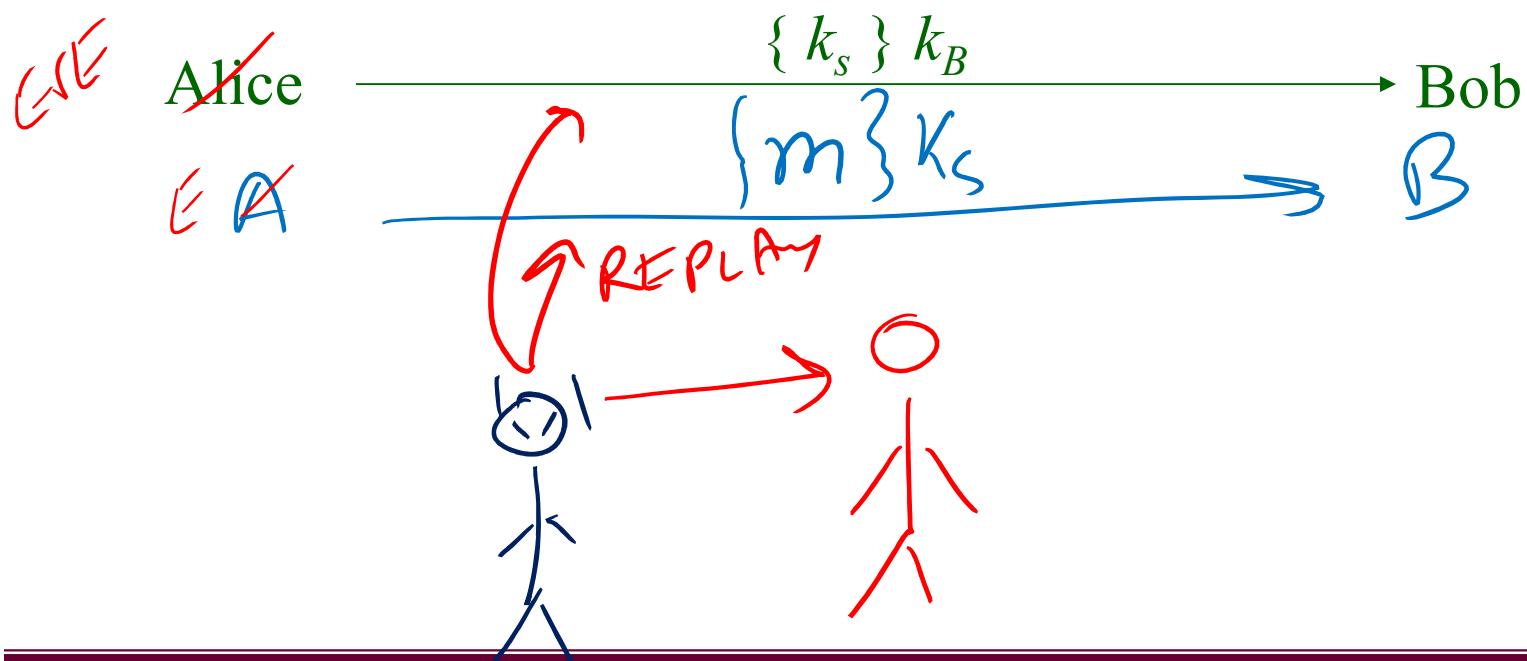
1 Alice $\xrightarrow{\{ \text{request for session key to Bob} \} k_A}$ Cathy

2 Alice $\xleftarrow{\{ k_s \} k_A \parallel \{ k_s \} k_B}$ Cathy

3 Alice $\xrightarrow{\{ k_s \} k_B}$ Bob

A $\xrightarrow{\{ m \} k_s}$ B

Problem with Simple Protocol



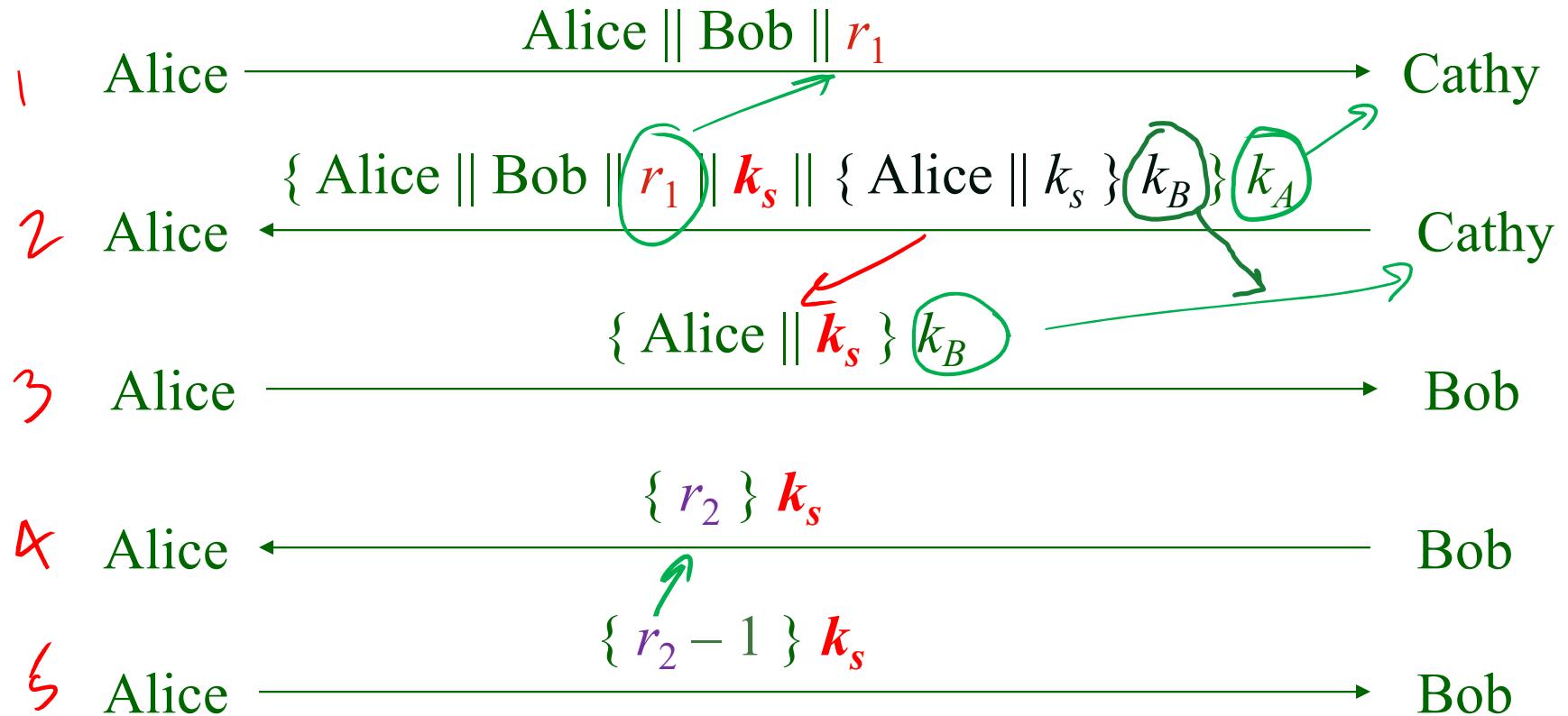
Concern

Replay attack

- Eve intercepts message from Alice to Bob, later replays it; Bob may think he's talking to Alice, but he isn't

Protocols must provide defense against replay

Needham-Schroeder Protocol



Assumptions

- r_1 and r_2 are two numbers generated randomly that are unique for each protocol exchange
- Both Alice and Bob trust Cathy
- Secret keys are secret !!

K_A K_B

A diagram consisting of two green arrows. One arrow points from the word 'secret' in the third bullet point to the label K_A . The other arrow points from the word 'secret' to the label K_B .

Argument: Alice talking to Cathy

□ Second message

- Enciphered using key only she, Cathy knows K_A
 - So Cathy enciphered it
- Response to first message
 - As r_1 in it matches r_1 in first message

Argument: Alice talking to Bob

- Third and fourth messages
 - Alice knows only Bob can read it
 - As only Bob can derive session key from message
 - Any messages enciphered with that key are from Bob

SALICE || K_s} } K_B

The handwritten note shows the string "SALICE || Ks" followed by two curly braces. The first brace groups the string "SALICE || Ks". The second brace groups the character "Ks" and the character "KB". Two green arrows point upwards from the bottom of each brace towards the respective groups.

Argument: Bob talking to Cathy

□ Third message

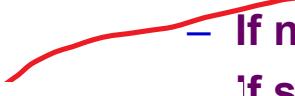
- Enciphered using key only he, Cathy know
 - So Cathy enciphered it
- Names Alice, session key
 - Cathy provided session key, says Alice is other party

SALICE || K_S} } K_B

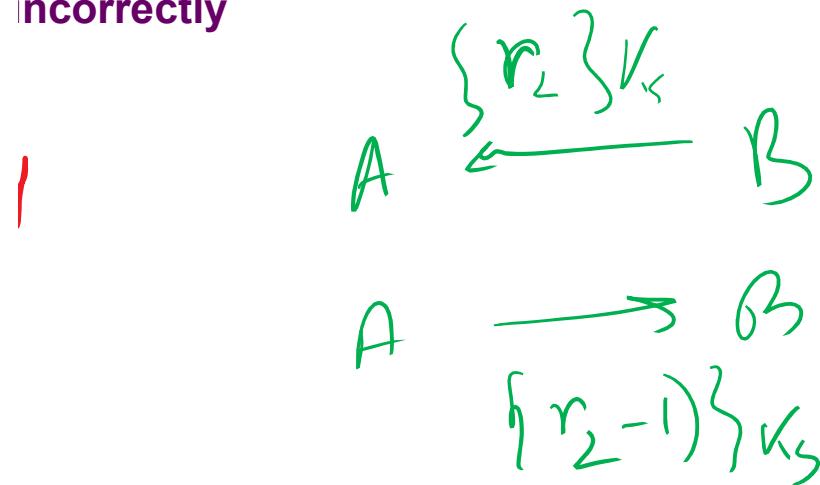
Argument: Bob talking to Alice

□ Fourth message and Fifth message

- Uses session key and nonce to determine if it is replay from Eve

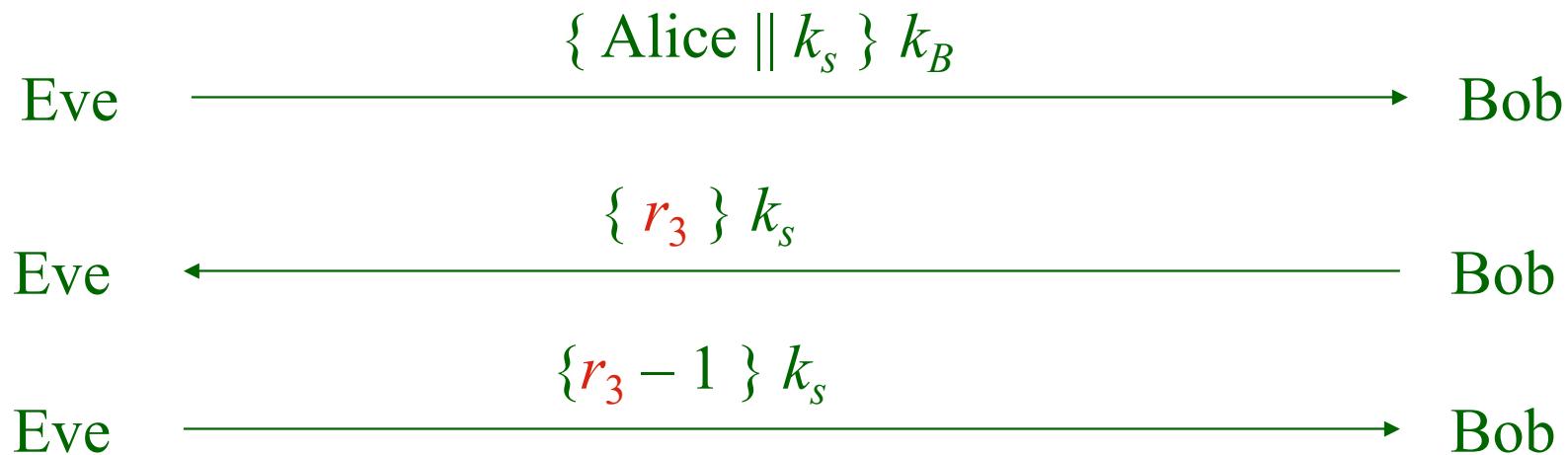
 If not, Alice will respond correctly in fifth message

If so, Eve can't decipher r_2 and so can't respond, or responds incorrectly



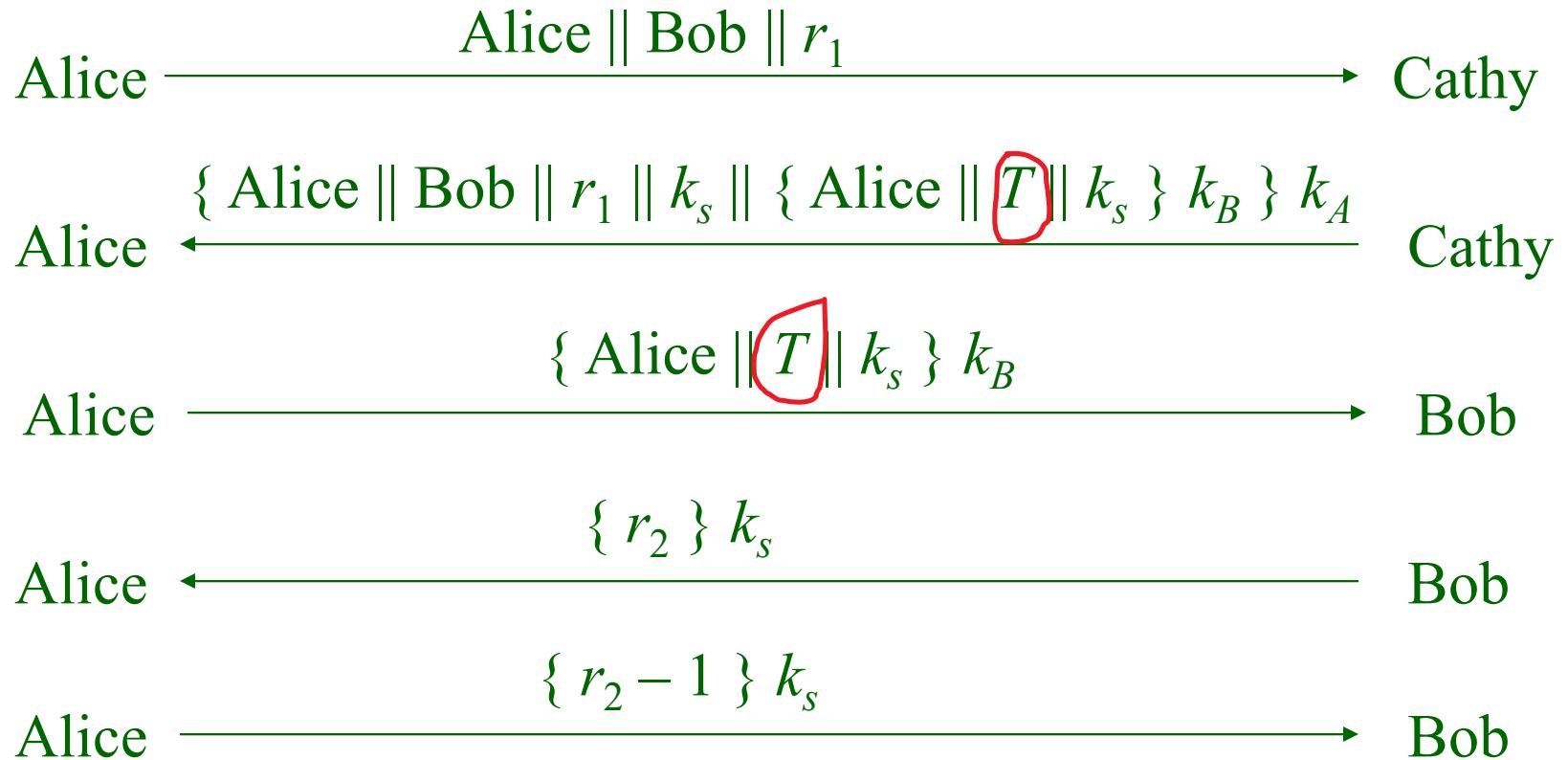
Problem with Needham-Schroeder Protocol

- What if Eve compromised the session key. How does that affect protocol?
 - In what follows, Eve knows recycled k_s

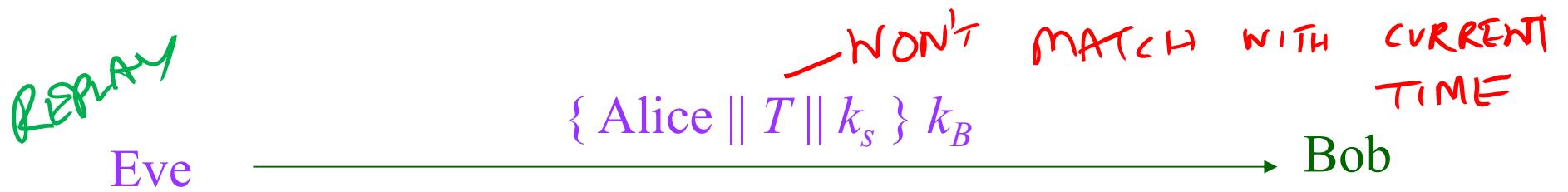


Needham-Schroeder with Denning-Sacco Modification

- Solution: use time stamp to detect replay



Good News



Bad news

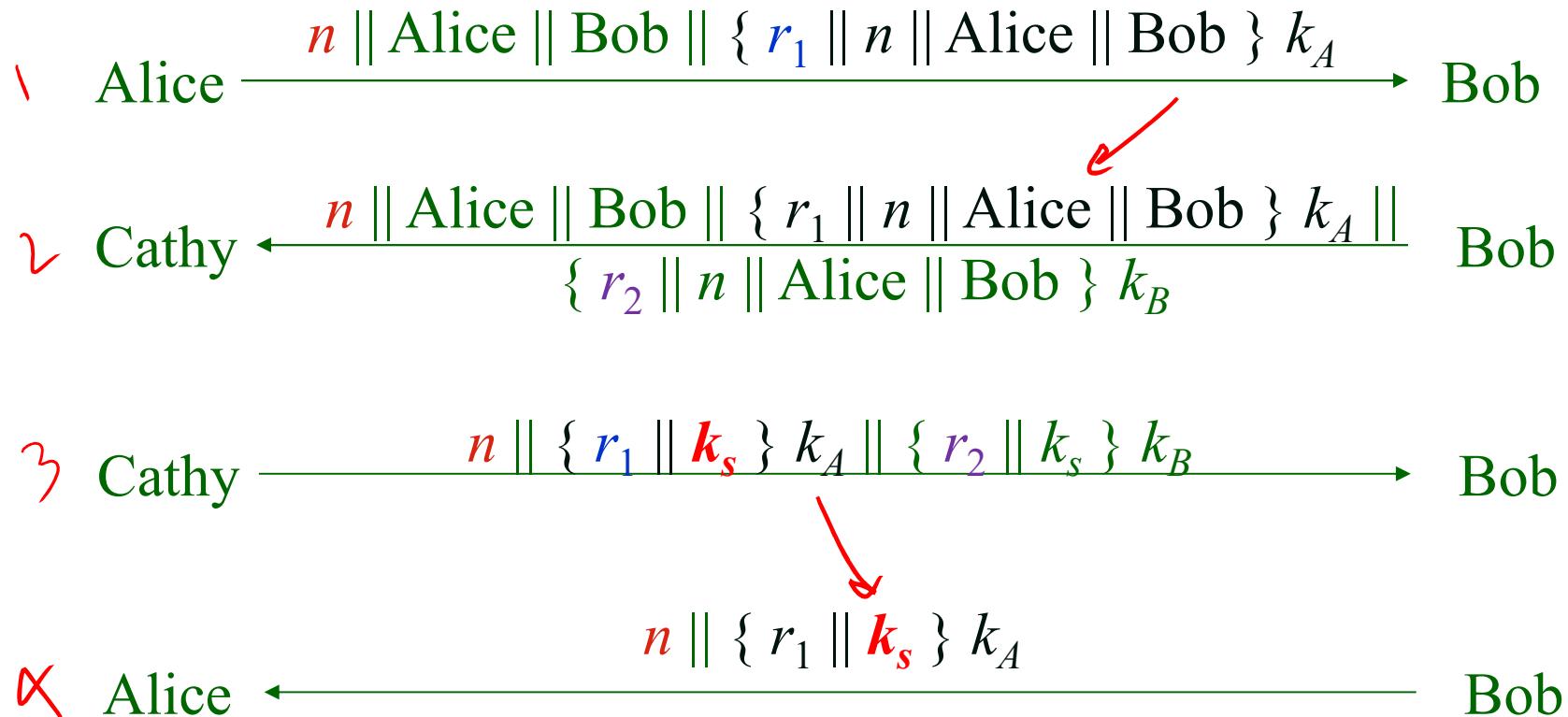
- If clocks not synchronized for communicating parties, may either reject valid messages or accept replays
 - Parties with either slow or fast clocks vulnerable to replay

TO REEP UP MTH
SAME T =

Otway-Rees (OR) Protocol

- Does not use timestamps to prevent replay
 - Not vulnerable to the problems that Denning-Sacco modification has
- Uses integer n to associate all messages with particular exchange

The OR Protocol



Argument: Alice talking to Bob

□ Fourth message

- If n matches first message, Alice knows it is part of this protocol exchange
- Cathy generated k_s because only she, Alice know k_A
- Enciphered part belongs to exchange as r_1 matches r_1 in encrypted part of first message

$$n \parallel \{ r_1 \parallel k_s \} k_A$$

Argument: Bob talking to Alice

□ Third message

- If n matches second message, Bob knows it is part of this protocol exchange
- Cathy generated k_s because only she, Bob know k_B
- Enciphered part belongs to exchange as r_2 matches r_2 in encrypted part of second message

$$n \parallel - - - - \parallel \{ r_2 \parallel k_s \} k_B$$

Replay Attack with Otway-Rees Protocol

- Eve acquires old k_s message in third step

- $n \parallel \{r_1 \parallel k_s\} k_A \parallel \{r_2 \parallel k_s\} k_B$

- Eve forwards appropriate part to Alice

- Good News

- Alice has no ongoing key exchange ; n matches nothing, so is rejected
 - Alice has ongoing key exchange; n does not match, so is again rejected

- Bad News

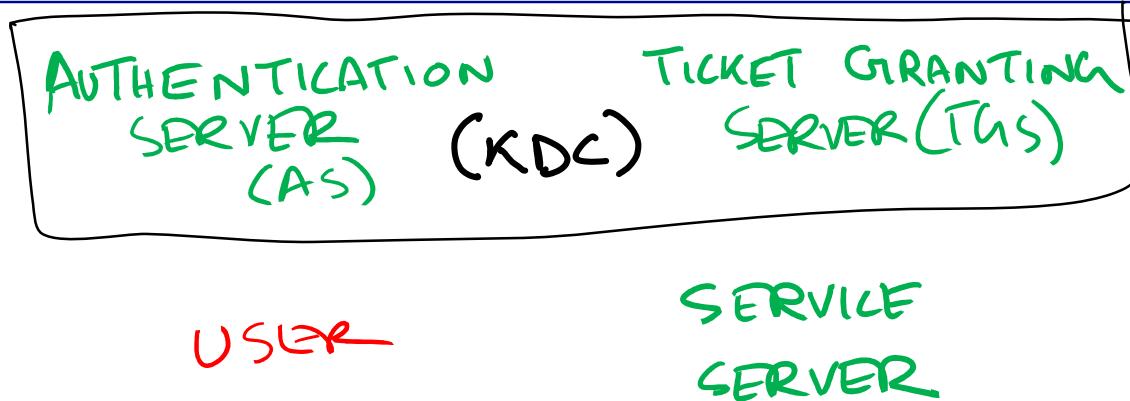
- IF replay is for the current key exchange, AND Eve sent the relevant part BEFORE the other party, only then it would work
 - Not so bad then



Kerberos

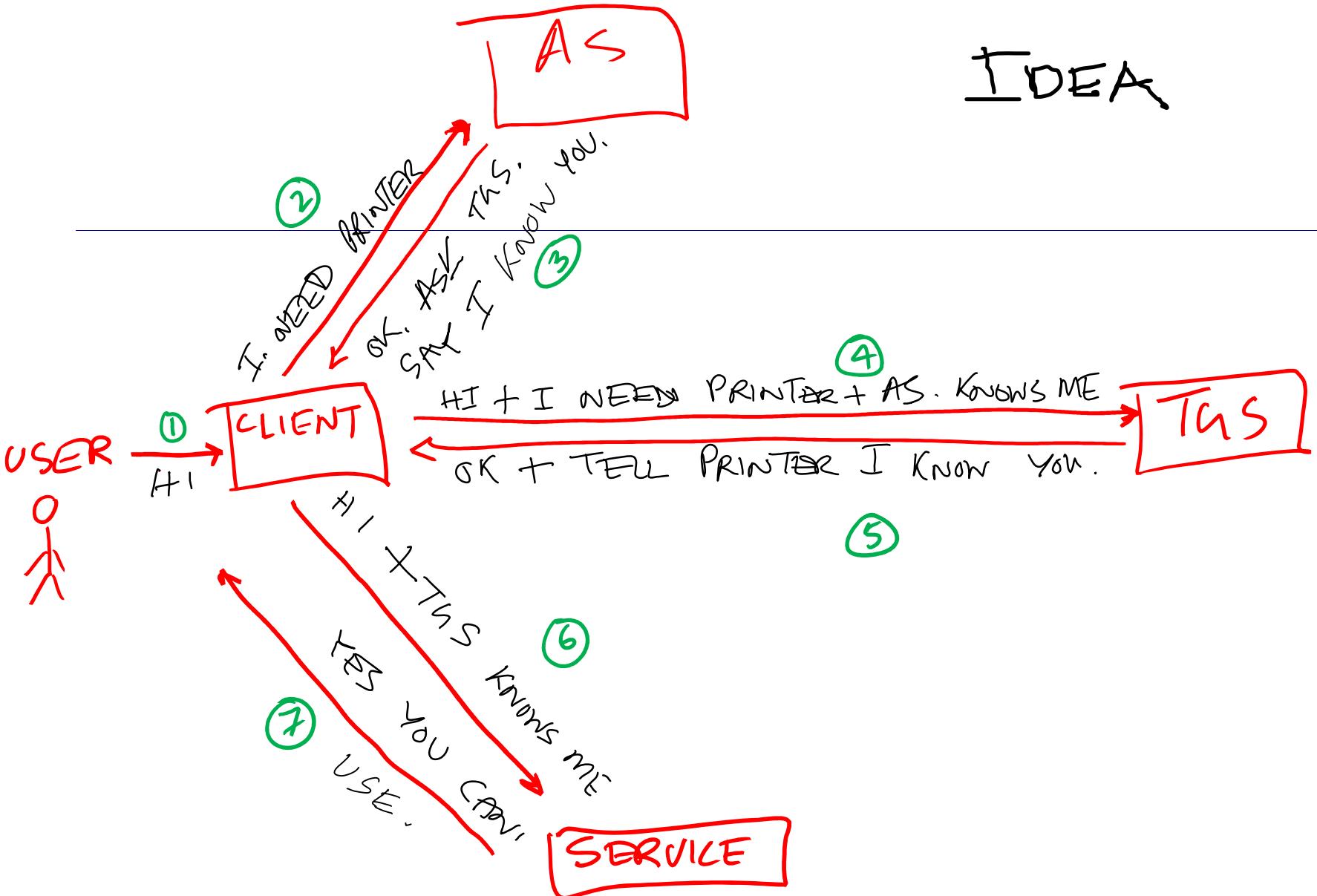
- Network Authentication Protocol for client-server systems
 - Created by MIT
 - Mutual authentication between user and service
 - Single sign-on
- Message encryption uses symmetric key cryptography with DES (version 4)
 - Extended version (version 5) allows AES and public key cryptography for authentication
- Key exchange based on Needham-Schroeder with Denning-Sacco modification

Concepts



❑ Key Distribution Center (KDC)

- Trusted third party (“Cathy”)
- Consists of
 - Authentication Server (AS)
 - Identifies requester of service
 - Ticket Granting Server (TGS)
 - Vouches for requester of service to the service



Kerberos Keys

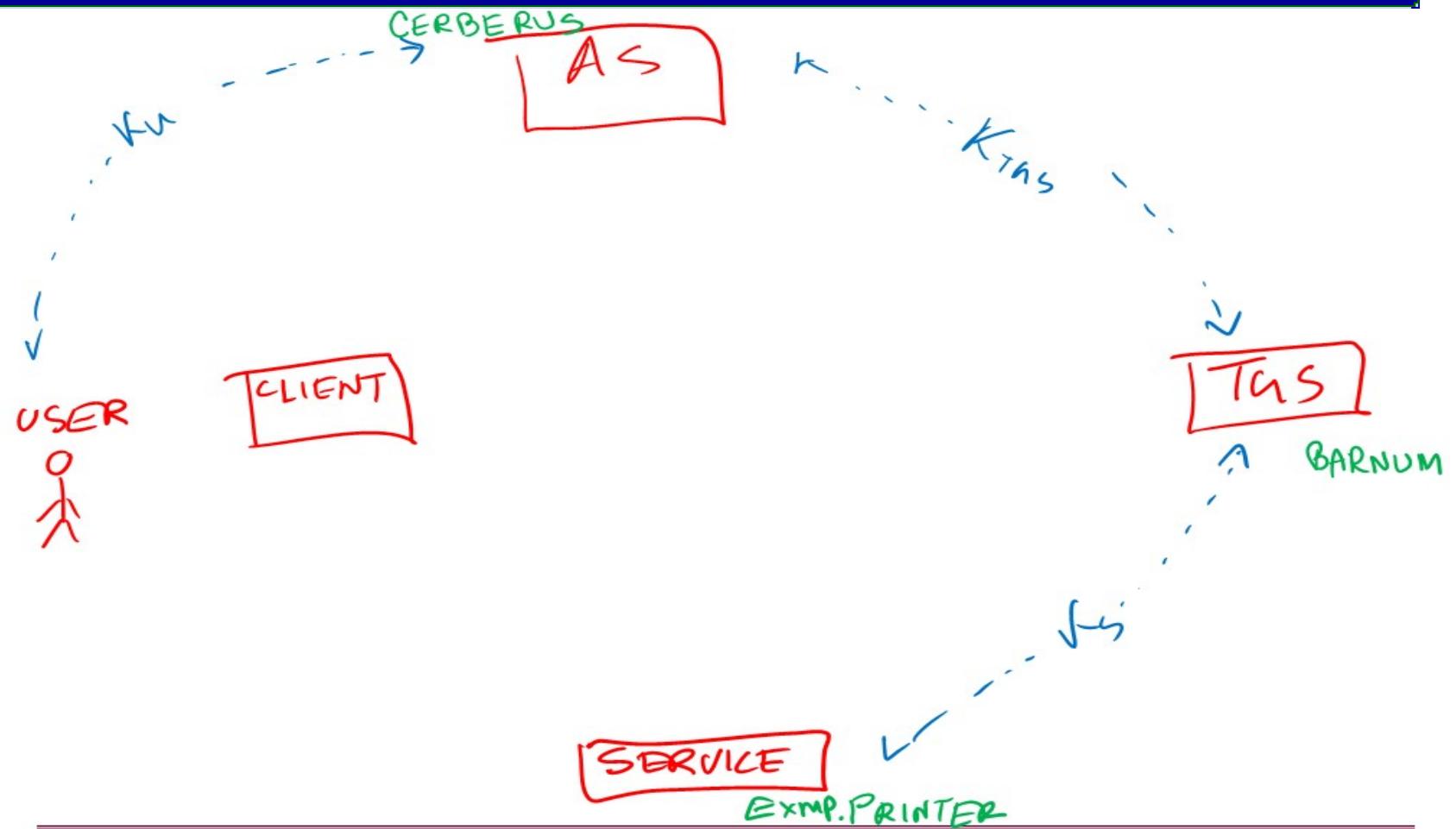
❑ Interchange keys

- Used to encrypt Tickets containing session keys
 - One shared between User and AS (user key) - k_u
 - One shared between AS and TGS - k_{TGS}
 - One shared between TGS and service S - k_s

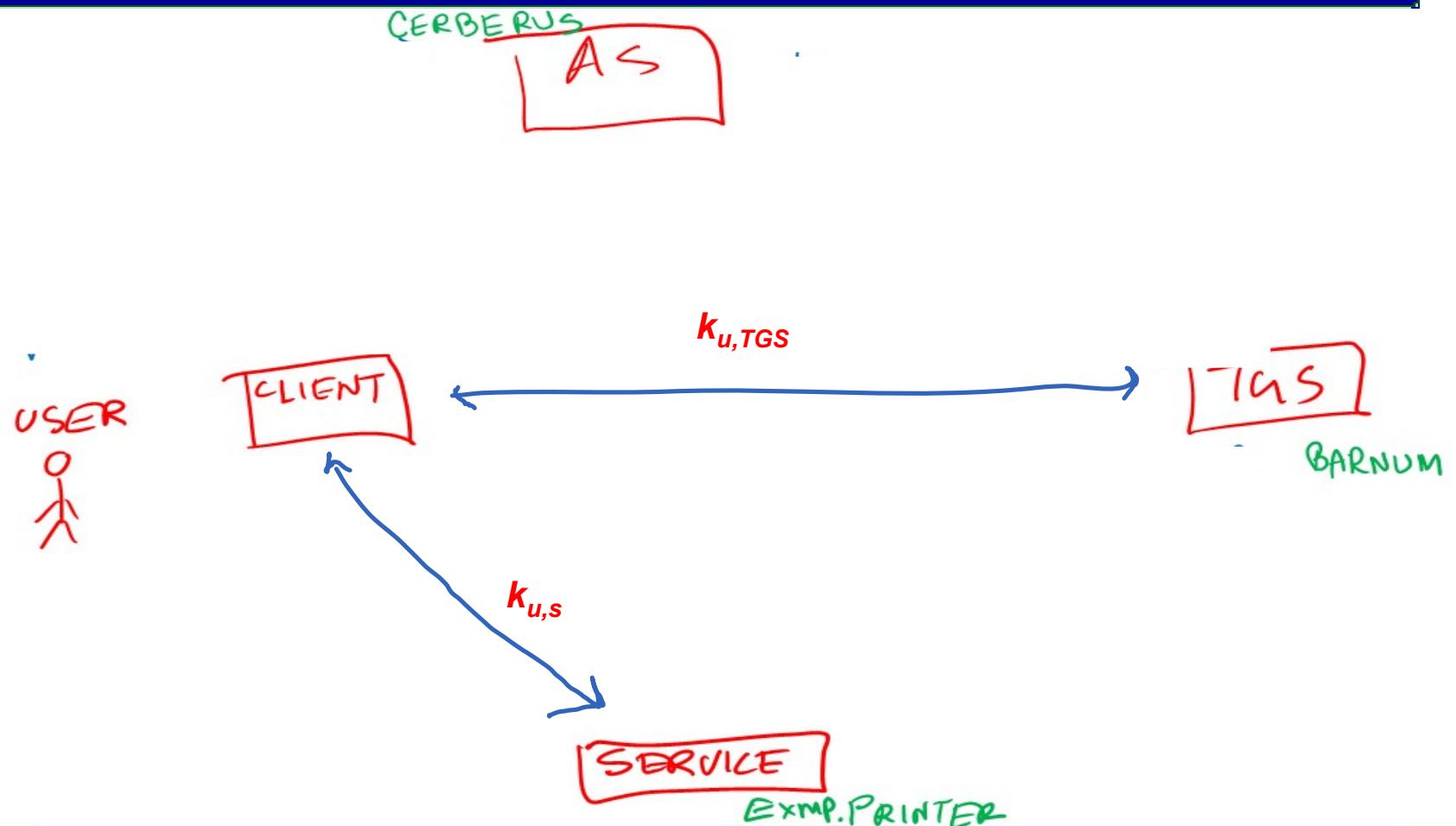
❑ Session keys

- Used to encrypt messages/authenticators
 - One shared between User and TGS- $k_{u,TGS}$
 - One shared between User and service S – $k_{u,s}$

Interchange Keys



Session Keys



Kerberos Credentials

- Authenticator** (LIKE DRIVING LICENSE)
- Ticket** (LIKE CONCERT TICKET, PLANE TICKET)

Kerberos Authenticator

Authenticator

- Credential containing identity of submitter of ticket
- Encrypted with session key that vouches for user's identity
- Generated by User (to show to TGS and service)

Authenticator for TGS

$$A_{u,TGS} = \{ u \parallel \text{generation time} \parallel k_t \} k_{u,TGS}$$

where:

- Session key $k_{u,TGS}$ for user and TGS
- k_t is alternate session key
- Generation time is when authenticator generated
 - Note: more fields, not relevant here

Authenticator for service s

$$A_{u,s} = \{ u \parallel \text{generation time} \parallel k_t \} k_{u,s}$$

where:

- Session key $k_{u,s}$ for user and service

Kerberos Ticket

□ Ticket

- Credential to serve as permission and shows that issuer (trusted 3rd party) has authenticated entity
 - Generated by AS (for TGS use) and by TGS (for service use)
- Distributes session key
 - Encrypted with interchange key
- Unforgeable, non-replayable, authenticated token

Kerberos Tickets

- Ticket issued to user u by AS for TGS (Ticket Granting Ticket/TGT)

$$T_{u,TGS} = \underline{\text{TGS}} \parallel \{ u \parallel u\text{'s address} \parallel \text{valid time} \parallel k_{u,TGS} \} \underline{k_{TGS}}$$

where:

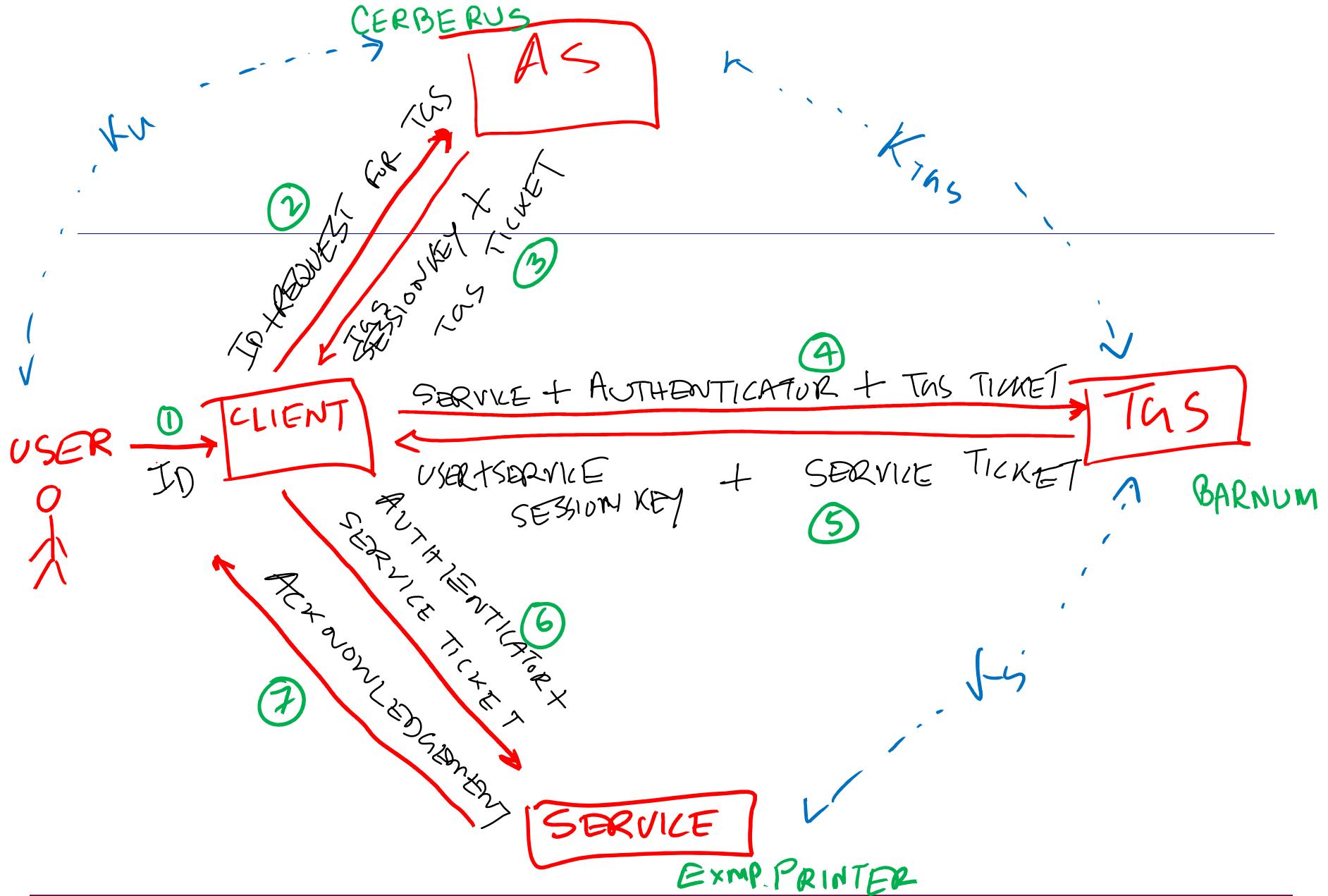
- $k_{u,TGS}$ is session key for user and TGS
- k_{TGS} is secret key shared by TGS and Authenticator
- Valid time is interval for which ticket valid
- u 's address may be IP address or something else

- Ticket issued to user u by TGS for service s

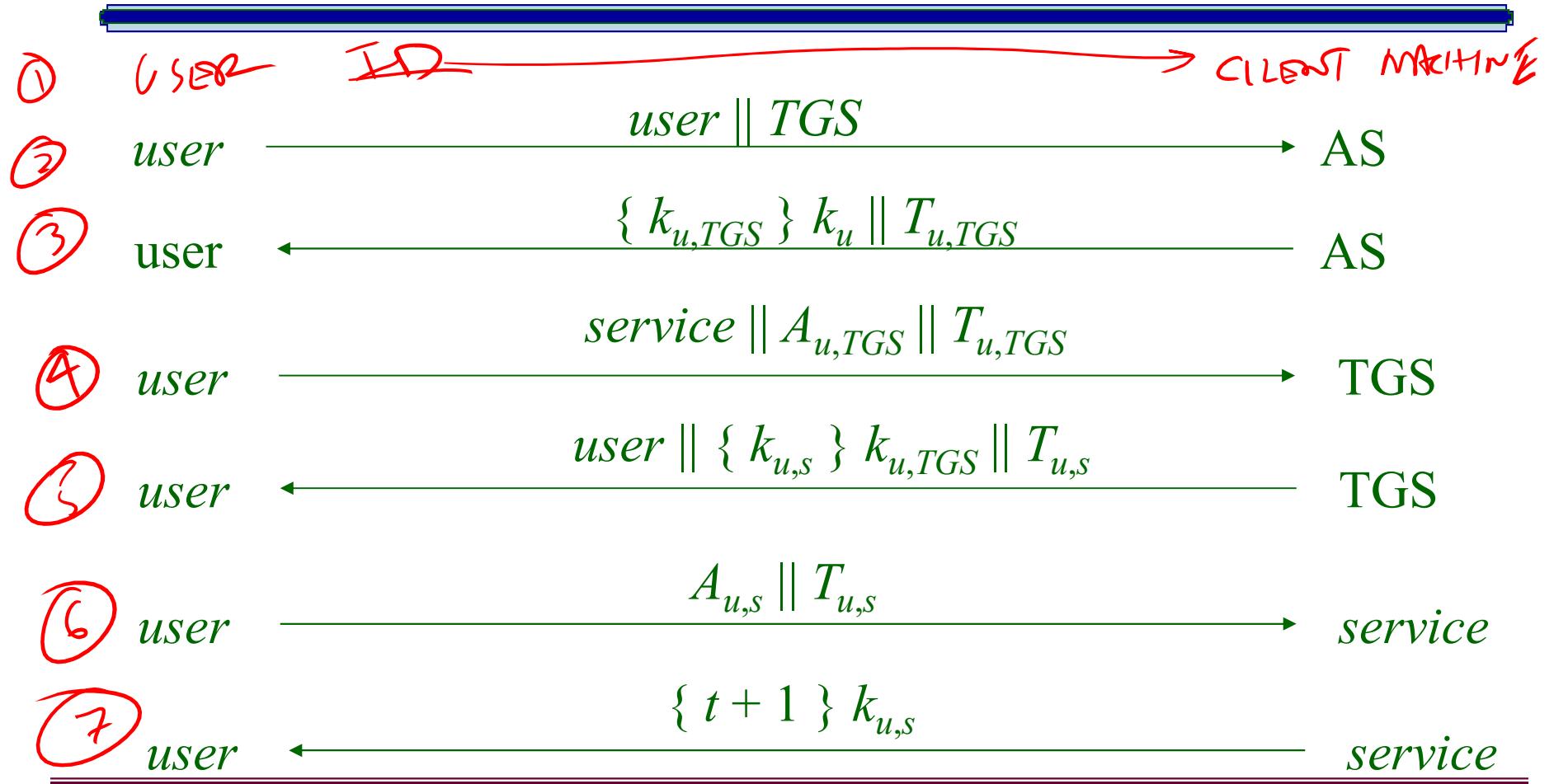
$$T_{u,s} = \underline{\text{service}} \parallel \{ u \parallel u\text{'s address} \parallel \text{valid time} \parallel k_{u,s} \} \underline{k_s}$$

where:

- $k_{u,s}$ is session key for user and service
- k_s is secret key shared by TGS and service



Protocol



Overview

- User u authenticates to Kerberos authentication server
 - Obtains ticket $T_{u,TGS}$ for ticket granting service (TGS)

- User u wants to use service s :
 - User generates authenticator A_u and sends it with ticket $T_{u,TGS}$ to TGS asking for ticket for service
 - TGS sends ticket $T_{u,s}$ to user
 - User sends $A_u, T_{u,s}$ to server as request to use s

Pros

- ✓ **Cryptographic protection against spoofing/eavesdropping**
 - ✓ **Time validity protection against long term attacks**
 - ✓ **Timestamp protection to prevent replay attacks**
 - ✓ **No password transmission**
-

Cons

- ✗ **Single point of failure**
- ✗ **Requires**
 - ✗ continuous availability of trusted TGS
 - ✗ trust relationship between TGS and each server
 - ✗ timely transactions
 - ✗ Relies on synchronized clocks
- ✗ **User workstation can be compromised and password logged and later replayed**
 - ✗ Password guessing can be a problem
- ✗ **Does not scale well**

Evolved Kerberos Version 5 <http://web.mit.edu/kerberos/>

Kerberos 5

Evolved Kerberos Version 5 <http://web.mit.edu/kerberos/>

- Internet standard (specified in RFC1510)
- Supports any symmetric key algorithm (not just DES)
- Longer ticket validity period
- Allows ticket to be renewed
- Supports different types of networking protocols (not just IP)
- Better protection against replay attacks
- Supports cross-realm (realm is set of nodes one KDC serves) authentication
- Continued support

Limited export restrictions <http://web.mit.edu/kerberos/dist/index.html>

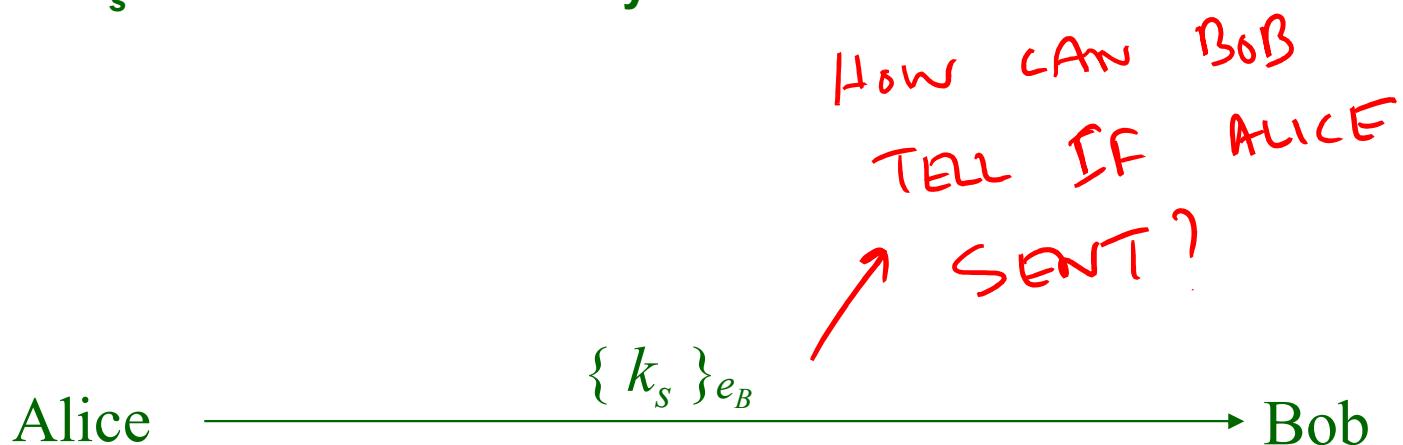
Public Key Key Exchange

Interchange keys

- e_A, e_B Alice and Bob's public keys known to all
- d_A, d_B Alice and Bob's private keys known only to owner

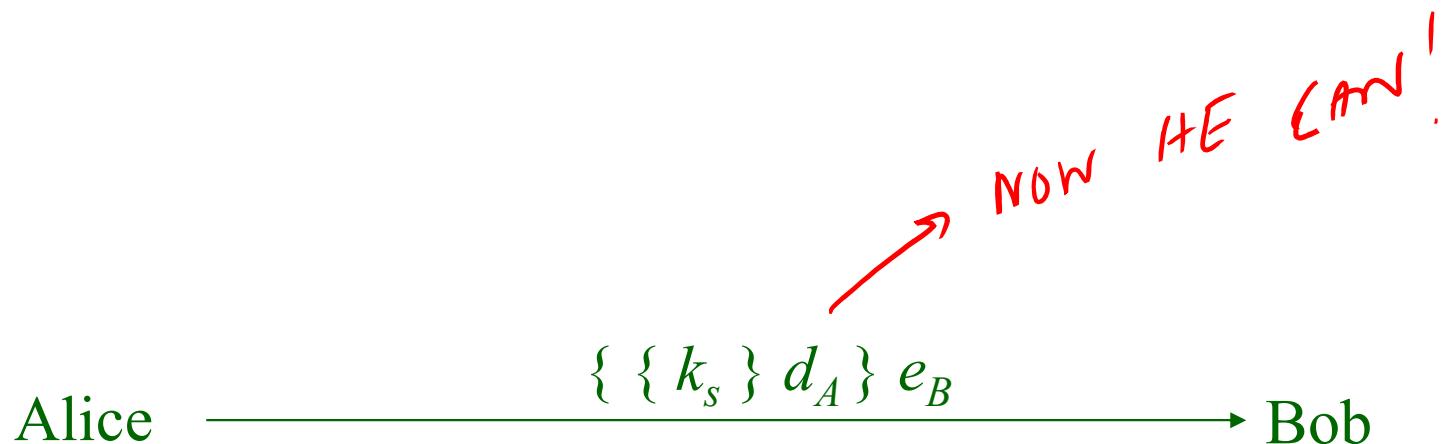
Simple protocol

- k_s is desired session key



Problem and solution

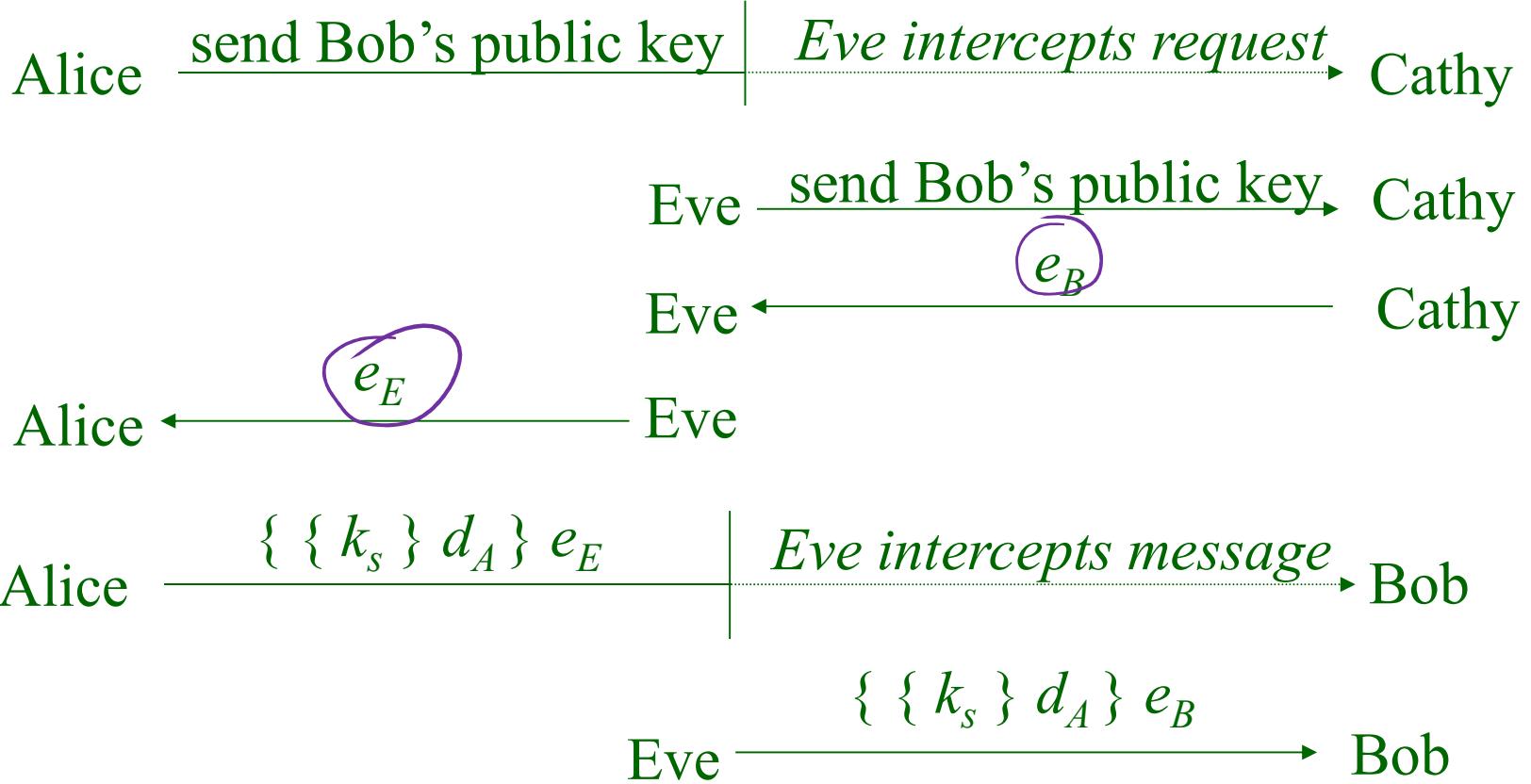
- Vulnerable to forgery or replay
 - e_B is known to everyone, Bob has no assurance that Alice sent message
- Simple fix using Alice's private key



But..

- Assumes Bob has Alice's public key, and *vice versa*
- If not, each must get it from public server
 - If keys not bound to identity of owner, attacker Eve can launch a *man-in-the-middle attack*
 - Attackers intercepts and relays messages between two parties pretending to be the “other” party to each one

Man-in-the-Middle Attack



Cryptographic Key Infrastructure

□ Binding of key to owner

- Classical cryptography: not applicable as all are shared secret keys
 - Use protocols to agree on a shared key
- Public key cryptography
 - Uses certificates to bind principal/user (identified by an acceptable name) to public key

Certificates

- ❑ Verifiable electronic notarized endorsement
 - Vouched/signed by trusted 3rd party
- ❑ Token (message) containing
 - Identity of principal (here, Alice)
 - Corresponding public key
 - Timestamp (when issued)
 - Other information (perhaps identity of signer/issuer)
- ❑ Protected against alteration
 - Includes checksum/hash

$$C_A = e_A \parallel \text{Alice} \parallel T \parallel \{h(e_A \parallel \text{Alice} \parallel T)\}$$

DATA

HASH

SIGNED

Using Certificates

Issuer Cathy

- Creates certificate
 - Generates hash of certificate
 - Signs hash with issuer's private key

$$C_A = e_A \parallel \text{Alice} \parallel T \parallel \{h(e_A \parallel \text{Alice} \parallel T)\} d_C$$

Requester Bob of Alice's public key

- Uses certificate
 - Get data
 - Obtains Alice's public key from certificate
 - Validate data
 - Obtains issuer's (Cathy's) public key
 - Deciphers hash
 - Recomputes hash from certificate
 - Compares hash and validates

Problem and Solution

- Bob needs Cathy's public key to validate certificate
 - Problem pushed "up" a level
- Solution:
 - ✓ Use of Certification Authorities (CA) who generates certificates
- Certificates
 - X.509 and OpenPGP

$$C_A = e_A \parallel \text{Alice} \parallel T \parallel \{h(e_A \parallel \text{Alice} \parallel T)\} d_C$$

X.509 Certificate

- A.k.a. Directory authentication framework
- Primary certificate components in X.509v3:

- Certificate
 - Version
 - Serial number
 - Signature encryption algorithm info
 - Issuer's name
 - Interval of validity
 - Subject's name
 - Subject's public key info
 - • Algorithm and key
- Signature
 - Hash algorithm
 - Enciphered hash

```

Certificate:
Data:
Version: 1 (0x0)
Serial Number: 7829 (0xe95)
Signature Algorithm: md5WithRSAEncryption
Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
        OU=Certification Services Division,
        CN=Thawte Server CA/emailAddress=server-certs@thawte.com
Validity
    Not Before: Jul 9 16:04:02 1998 GMT
    Not After : Jul 9 16:04:02 1999 GMT
Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
        OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (1024 bit)
        Modulus (1024 bit):
            00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
            33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
            66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
            70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
            16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
            c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
            8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
            d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
            e8:35:1c:9e:27:52:7e:41:8f
        Exponent: 65537 (0x10001)
Signature Algorithm: md5WithRSAEncryption
93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:lb:41:72:
0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:b3:f3:d9:f7:
8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
68:9f

```

<http://en.wikipedia.org/wiki/X.509>

CS 4575/5575 Information Assurance and Security

PKI Models

Public Key Infrastructure (PKI)

- Management system that allows the creation, distribution and revocation of X.509 certificates through certificate authorities

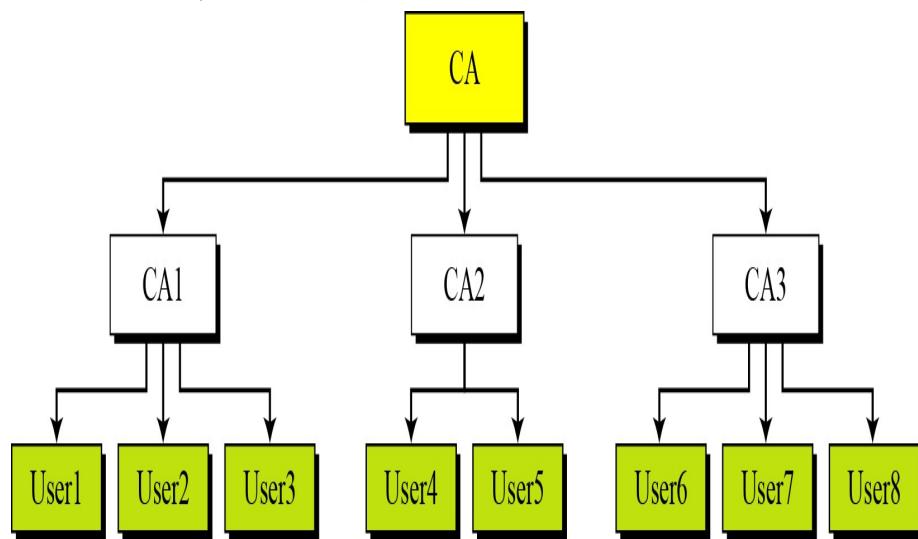
Trust Models

- Hierarchical CAs with root CA who can self sign
- Mesh model with cross-certification (e.g. X.509)
 - Disjoint hierarchical groups with root CAs who cross certify

```
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 1 (0x1)
Signature Algorithm: md5WithRSAEncryption
Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
OU=Certification Services Division,
CN=Thawte Server CA/emailAddress=server-certs@thawte.com
Validity
Not Before: Aug 1 00:00:00 1996 GMT
Not After : Dec 31 23:59:59 2020 GMT
Subject: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
OU=Certification Services Division,
CN=Thawte Server CA/emailAddress=server-certs@thawte.com
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
00:d3:a4:50:6e:c8:ff:56:6b:e6:cf:5d:b6:ea:0c:
68:75:47:a2:aa:c2:da:84:25:fc:a8:f4:47:51:da:
85:b5:20:74:94:86:1e:0f:75:c9:e9:08:61:f5:06:
6d:30:6e:15:19:02:e9:52:c0:62:db:4d:99:9e:e2:
6a:0c:44:38:cd:fe:be:3d:64:09:70:c5:fe:b1:6b:
29:b6:2f:49:c8:3b:d4:27:04:25:10:97:2f:e7:90:
6d:c0:28:42:99:d7:4c:43:de:c3:f5:21:6d:54:9f:
5d:c3:58:e1:c0:e4:d9:5b:b0:b8:dc:b4:7b:df:36:
3a:c2:b5:66:22:12:d6:67:0d
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Basic Constraints: critical
CA:TRUE
Signature Algorithm: md5WithRSAEncryption
07:fa:4c:69:5c:fb:95:cc:46:ee:65:63:4d:21:30:6e:ca:d9:
a8:6f:49:1a:e6:da:51:e3:60:70:6c:84:61:11:a1:1a:c8:48:
3e:59:43:7d:4f:95:3d:a1:8b:b7:0b:62:98:7a:75:8a:dd:88:
4e:4e:9e:40:db:a8:cc:32:74:b9:6f:0d:c6:e3:b3:44:0b:d9:
8a:6f:9a:29:9b:99:18:28:3b:d1:e3:40:28:9a:5a:3c:d5:b5:
e7:20:1b:9b:ca:a4:ab:8d:e9:51:d9:a2:4c:2c:59:a9:da:b9:
b2:75:1b:f6:42:f2:ef:c7:r2:18:t9:89:bc:a3:ff:8a:23:2e:
70:47
```

PKI Models (Another View)

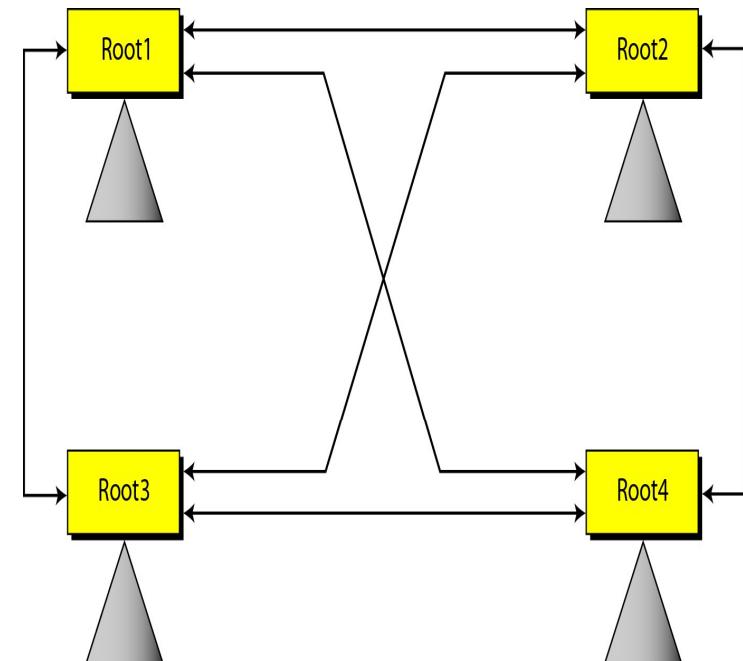
HIERARCHICAL



$X \rightarrow Y$

means X has signed a certificate for Y

MESH



$X \leftrightarrow Y$

means X and Y have signed certificates for each other.

Cross-Certifying in X.509

- Arbitrary length certificate chains rely on individual's knowledge of certifiers

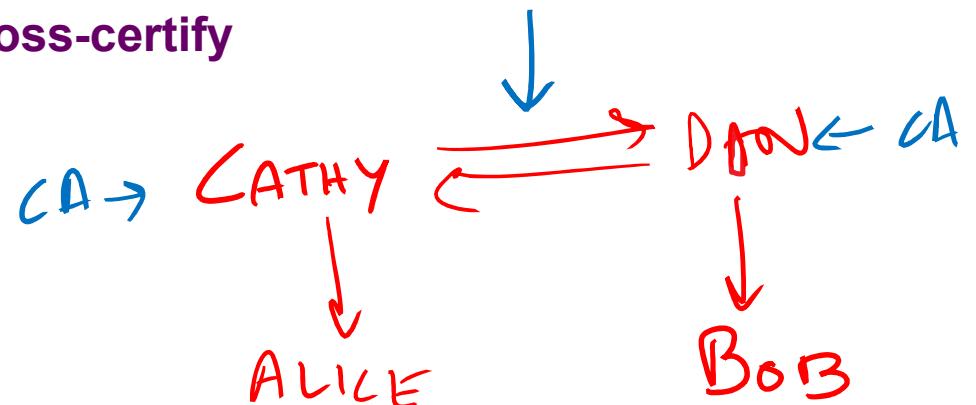
- Alice's CA is Cathy; Bob's CA is Don;
how can Alice validate Bob's certificate?
 - Have Cathy and Don cross-certify

- Certificates:

- Cathy<<Alice>>
 - Don<<Bob>>
 - Cathy<<Don>>
 - Don<<Cathy>>

- Alice validates Bob's certificate

- Alice uses (known) public key of Cathy to validate
Cathy<<Don>>
 - Alice uses Cathy<<Don>> to validate Don<<Bob>>



OpenPGP

- ❑ Primary certificate standard for PGP (Pretty Good Privacy)
 - Used for secure email communication
- ❑ PGP initially supported only Web of Trust model where
 - Anyone can sign for anyone (even self sign)
 - No “authority”
 - Instead, users certify each other to build a “web of trust”
 - (Current version also allows hierarchical model with X.509 certificates)
- ❑ OpenPGP certificates structured into two types of packets
 - One public key packet
 - Zero or more signature packets

OpenPGP Public Key Packet

□ Public key packet:

- Version (3/4)
- Creation time → NOT IN X.509
- Validity period
- Public key algorithm, associated parameters
- Public key

OpenPGP Signature Packet

❑ Version 3 signature packet

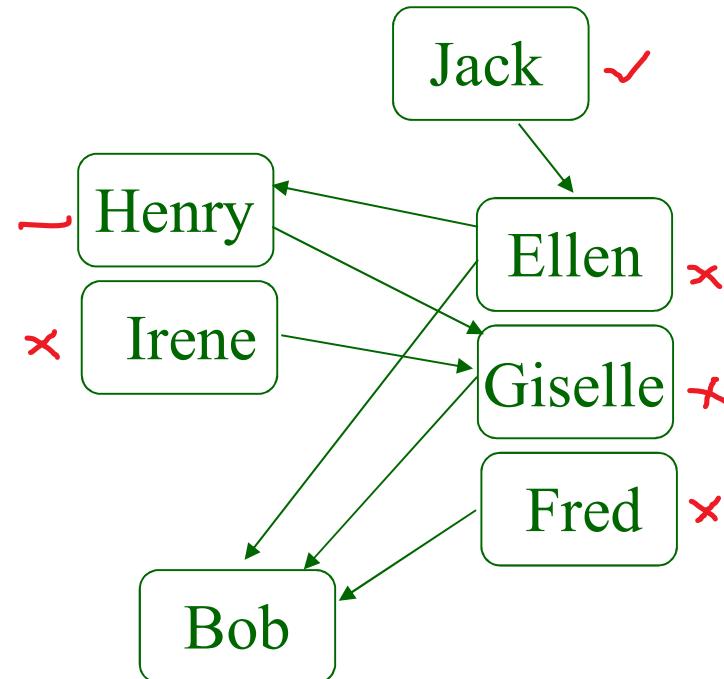
- Certificate
 - Version (3)
 - Signature type (level of trust)
 - Range from “untrusted” to “ultimate trust”
 - Creation time
 - Signer’s signature key identifier
 - Signer’s signature key algorithm
 - Hash algorithm used to generate the hash
- Part of signed hash (used for quick check)
- Signature (enciphered hash using signer’s private key)

❑ Version 4 packet more complex

Validating Certificates with PGP

- ❑ Alice needs to validate Bob's OpenPGP certificate
 - Does not know Fred, Giselle, or Ellen
- ❑ Alice gets Giselle's certificate
 - Does not know Irene
 - Knows Henry, but his signature is at "casual" level of trust
- ❑ Alice gets Ellen's certificate
 - Knows Jack who certifies at full trust, so uses his cert to validate Ellen's, then hers to validate Bob's

Arrows show signatures
Self signatures not shown



Storing Asymmetric Keys

- For public key
 - IA goal is integrity
 - Achieved with Certificates
- For private/secret key
 - IA goal is confidentiality
 - Multi-user or networked systems: attackers may defeat access control mechanisms
 - Encipher file containing key
 - Attacker can monitor keystrokes to decipher files
 - Key will be resident in memory that attacker may be able to read
 - Use physical devices like “smart card”
 - Key never enters system
 - Card can be stolen, so have 2 devices combine bits to make single key

Revoking Keys

- Certificates invalidated before expiration
 - Usually due to compromised key/issuer
 - May be due to change in circumstance (e.g., someone leaving company)
 - Problems with revoking
 - Circulating revocation information quickly
 - Checking if the entity revoking certificate is authorized to do so
- PRIVATE KEY of ISSUED SUBJECT

Certificate Revocation

- X.509: only certificate issuer can revoke certificate
 - Added to CRL
 - *Certificate revocation list (CRL)* lists certificates that are revoked
 - Date and Serial
- OpenPGP: issuer, as well as, owner can revoke certificates, or allow others to do so
 - Revocation message placed in OpenPGP packet and signed
 - Expiry dates

~~Primary Differences~~ between X.509 and OpenPGP

X.509

(Single path from fully trusted authority to any certificate) (Multiple path from fully/partially trusted authority to any certificate)

- One certificate has one issuer
- No level of trust
- Hierarchical (with root CA) model
- Greater impact if issuer (CA) is compromised
- Only issuer can revoke
- Use CRL for revocation

PGP

- A single certificate may have multiple signatures
- Signature type (level of trust)
- Web of trust model
- Lesser impact if issuer is compromised
- Both issuer and owner can revoke
- Use info within PGP packets

End Note

□ Security is difficult because

- Security is not inherent.
- Security is not universal.
- Security is not static.
- Security is not an absolute.

EXPECTATION



PROTECTION



PERCEPTION

□ Security is a compromise between usability, cost and peace of mind.

Confinement Problem

CSC 4575/5575
Information Assurance and Cryptography
Spring 2019

Sources:

Introduction to Computer Security, Matt Bishop, Addison Wesley, 2003
Security in Computing, Pfleeger, 2004

Overview

- ❑ The confinement problem
- ❑ Isolating entities
- ❑ Covert channels

Generalization of Client-Server Systems

- Client sends request, data to server
- Server performs some function on data
- Server returns result to client
- Access controls:
 1. Server must ensure the resources it accesses on behalf of client include *only* resources client is authorized to access
 2. Server must ensure it does not reveal client's data to any entity not authorized to see the client's data

Confinement Problem

- Problem of preventing a server from leaking information that the user of the service considers confidential

Total Isolation

- ❑ Process cannot communicate with any other process
- ❑ Process cannot be observed

Impossible for this process to leak information

- Not practical as process uses observable resources, such as CPU, secondary storage, networks, etc.

Example Problem1: Regarding Resource Sharing

- Processes *p*, *q* not allowed to communicate
 - But they share a file system!
 - Communications protocol:
 - *p* sends a bit by creating a file called *0* or *1*, then a second file called *send*
 - *p* waits until *send* is deleted before repeating to send another bit
 - *q* waits until file *send* exists, then looks for file *0* or *1*; whichever exists is the bit
 - *q* then deletes *0*, *1*, and *send* and waits until *send* is recreated before repeating to read another bit
 - *p* ends by creating a file called *end*
 - Covert Channel
-

Transitive Confinement

- ❑ If p is confined to prevent leaking, and it invokes q , then q can leak the information that p passes.
- ❑ Rule:
 - If a confined process invokes a second process, the second process must be as confined as the first.
- ❑ Confinement should be on the transmission as well.

Example Problem 2: Regarding Resource Sharing

- Another type of covert channel.
 - A process can obtain (“read”) rough idea of time
 - Counting the number of instructions executed during a period
 - System clock or wall clock
 - A process can manipulate (“write”) time
 - Executing a number of instructions and stopping, allowing another process to execute
 - This is possible when a process share the computer with another process
 - Isolation is a remedy.
-

Protection through Isolation

❑ Virtual machine

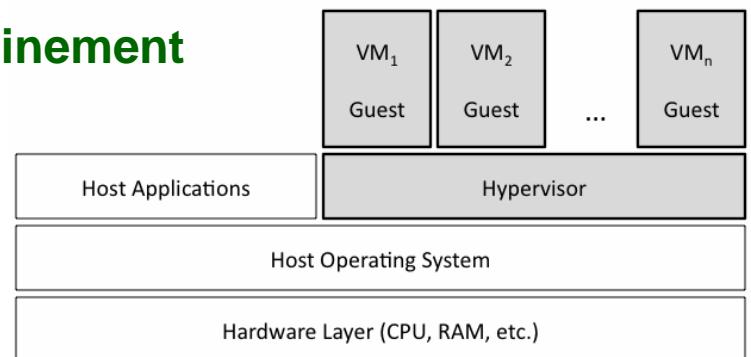
- Emulate computer
- Process cannot access underlying computer system, or anything that is not part of the environment

❑ Sandbox

- It is not meant to emulate computer
 - Virtual machine is a kind of sandboxing
- Alters interface between computer, process

Virtual Machine (VM)

- A program that simulates hardware of computer system
- Virtual Machine Monitor (VMM) or Hypervisor provides VM on which conventional OS can run
 - Each VM is one subject – VMM knows nothing about processes running on each VM
 - VMM mediates all interactions of VM with resources, other VMs
 - Enforces rule of transitive confinement



Sandbox

- ❑ Environment in which actions of process are restricted according to security policy
- ❑ Two ways
 - Limiting execution
 - Adding extra security-checking mechanisms to libraries, kernel
 - Program to be executed is not altered
 - Intervening
 - Modifying program or process to be executed
 - Similar to debuggers, profilers that add breakpoints
 - Add code to do extra checks (memory access, etc.) as program runs (*software fault isolation*)

Covert Channels

- A path of communication NOT intended for communication
 - Using shared resources as a communication path
- Covert storage channel uses attribute of shared resource
- Covert timing channel uses temporal or ordering relationship among accesses to shared resource

Example: Covert Storage Channel

- Shared file system
 - We have already seen this
 - Communications protocol:
 - *p* sends a bit by creating a file called *0* or *1*, then a second file called *send*
 - *p* waits until *send* is deleted before repeating to send another bit
 - *q* waits until file *send* exists, then looks for file *0* or *1*; whichever exists is the bit
 - *q* then deletes *0*, *1*, and *send* and waits until *send* is recreated before repeating to read another bit
 - Covert storage channel: resource is directory, names of files in directory
-

Example: Covert Timing Channel

Real-Time Clock

- We have seen this already

Virtual machines sharing the same physical device (e.g., KVM/370) can have a covert timing channel

- VM1 wants to send 1 bit to VM2
- To send 0 bit: VM1 relinquishes CPU as soon as it gets CPU
- To send 1 bit: VM1 uses CPU for full quantum
- VM2 determines which bit is sent by seeing how quickly it gets CPU
- Shared resource is CPU, timing because real-time clock used to measure intervals between accesses

Example: Ordering of Events

- A timing covert channel, not by real clock
 - Two VMs
 - Share cylinders 100–200 on a disk
 - One is *High*, one is *Low*; process on *High* VM wants to send to process on *Low* VM
 - Disk scheduler uses SCAN algorithm
 - *Low* process seeks to cylinder 150 and relinquishes CPU
 - *High* wants to send a bit
 - To send 1 bit, *High* seeks to cylinder 140 and relinquish CPU
 - To send 0 bit, *High* seeks to cylinder 160 and relinquish CPU
 - *Low* issues requests for tracks 139 and 161
 - Seek to 139 first indicates a 1 bit
 - Seek to 161 first indicates a 0 bit
 - Covert timing channel: uses ordering relationship among accesses to transmit information

Noiseless and Noisy Covert Channels

❑ Noiseless covert channel

- Uses shared resource available to sender, receiver only**

❑ Noisy covert channel

- Uses shared resource available to sender, receiver, and others**
- Attackers need to minimize interference enough so that message can be read in spite of others' use of channel**

Key Properties of Covert Channels

❑ Existence

- Determining whether covert channel can exist**

❑ Bandwidth

- Determining how much information can be sent over the channel**

Defense against Covert Channels

Identify Covert Channel

- Eliminate existence
- Reduce bandwidth

Detection

- Covert channels require sharing
- Manner of sharing controls
 - Which subjects can send
 - Which subjects can receive
- Ways of identification
 - Shared resource matrix
 - Covert flow trees

Shared Resource Matrix

Detection approach by Kemmerer

- Use matrix of resources (rows) and processes that can access them (columns)
- The entries denote the access rights
- Analyst examine the matrix to see if information can be leaked
- Example:

– Process Low is not allowed to read Resource 2

	Process High	Process Low
Resource 1	Read, Modify	Read, Modify
Resource 2	Read	

– Process Low CAN read Resource 2 with the help of Process High

	Process High	Process Low
Resource 1	Read, Modify	Read, Modify
Resource 2	Read	Read

Covert Flow Trees

□ Porras and Kemmerer

- Model “flow of information through shared resources” with a tree structure
 - Tree structured representation of sequence of operations that move information from one process to another
- Analyst examines the flow path to see if security is violated

Shared Resource Matrix vs. Covert Flow Tree

- Shared resource matrix can only identify existence of channels**

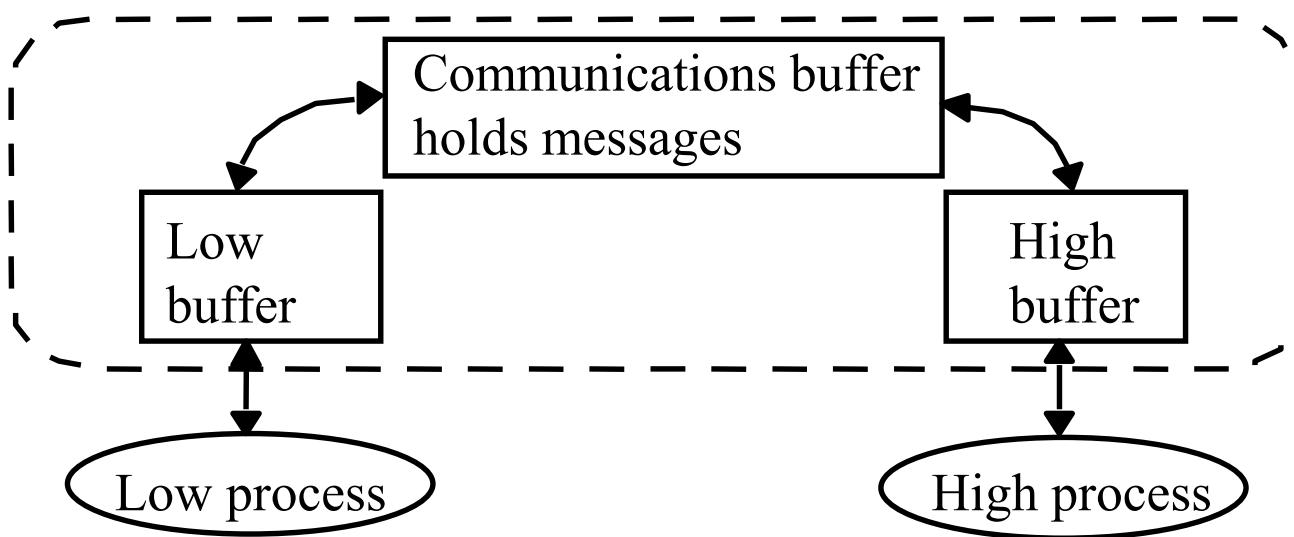
- Covert flow trees can identify existence of (more) channels and sequence of operations that the attacker can use for the covert channels**

Covert Channel Mitigation

- Obvious fix:
 - Ask processes what resources they need and provide them those ONLY
 - Need must specify the runtime as well
 - Take back after quota finishes

- Another way:
 - Obscure amount of resources a process uses
 - Receiver cannot determine what part sender is using and what not
 - How to do this?
 - Devote uniform, fixed amount of resources to each process
 - Inject randomness into allocation, use of resources
 - Consider tradeoff for loss in efficiency

Another Covert Channel: Pump



Problem

- ❑ Assume: Low process can process messages faster than High process
- ❑ Low process sends messages faster than High process can remove them
 - Opens covert channel

Covert Timing Channel with Pump

- ❑ High process can control rate at which pump sends it messages
- ❑ Initialization: Low sends messages to pump until communications buffer full
 - Low gets ACK for each message put into the buffer; no ACK for messages when communications buffer full
- ❑ Protocol: sequence of trials; for each trial
 - High sends a 1 by reading a message
 - Then Low gets ACK
 - High sends a 0 by not reading a message
 - Then Low doesn't get ACK

How to Fix

❑ Closing covert channel

- Making High process handle messages more quickly than Low process gets acknowledgements
 - Pump artificially delaying ACKs
 - Low cannot tell whether buffer is full
 - Not optimal (processes may wait even when unnecessary)

❑ Decreasing bandwidth

- Making pump and processes handle messages at same rate
 - Decreases bandwidth of covert channel
 - Sub-optimal performance

Key Points

- ❑ Confinement problem: prevent leakage of information
 - Solution: separation and/or isolation
- ❑ Shared resources offer paths along which information can be transferred
- ❑ Covert channels difficult if not impossible to eliminate
 - Bandwidth can be greatly reduced