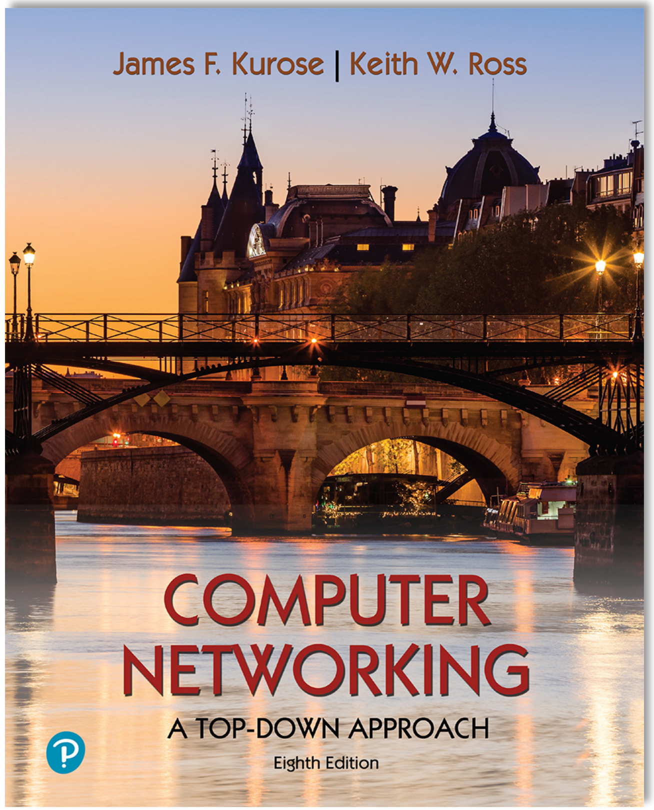
Chapter 8
Security



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

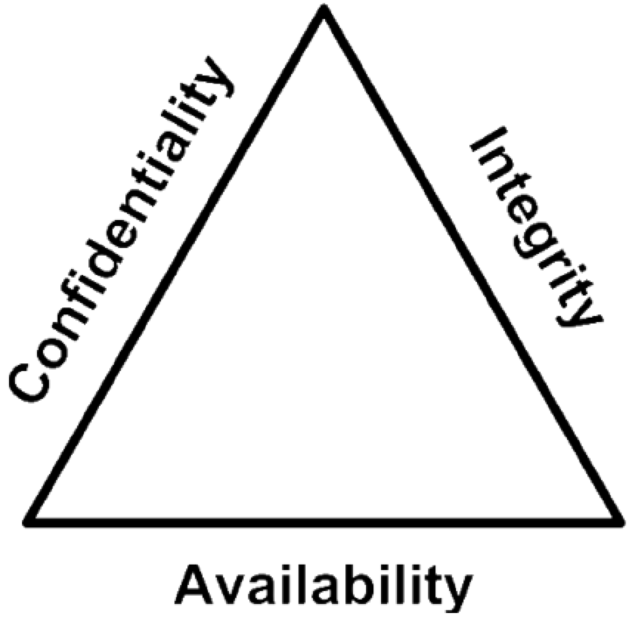
Acknowledgement: Based on the textbook's website:
https://gaia.cs.umass.edu/kurose_ross/index.php

Outline

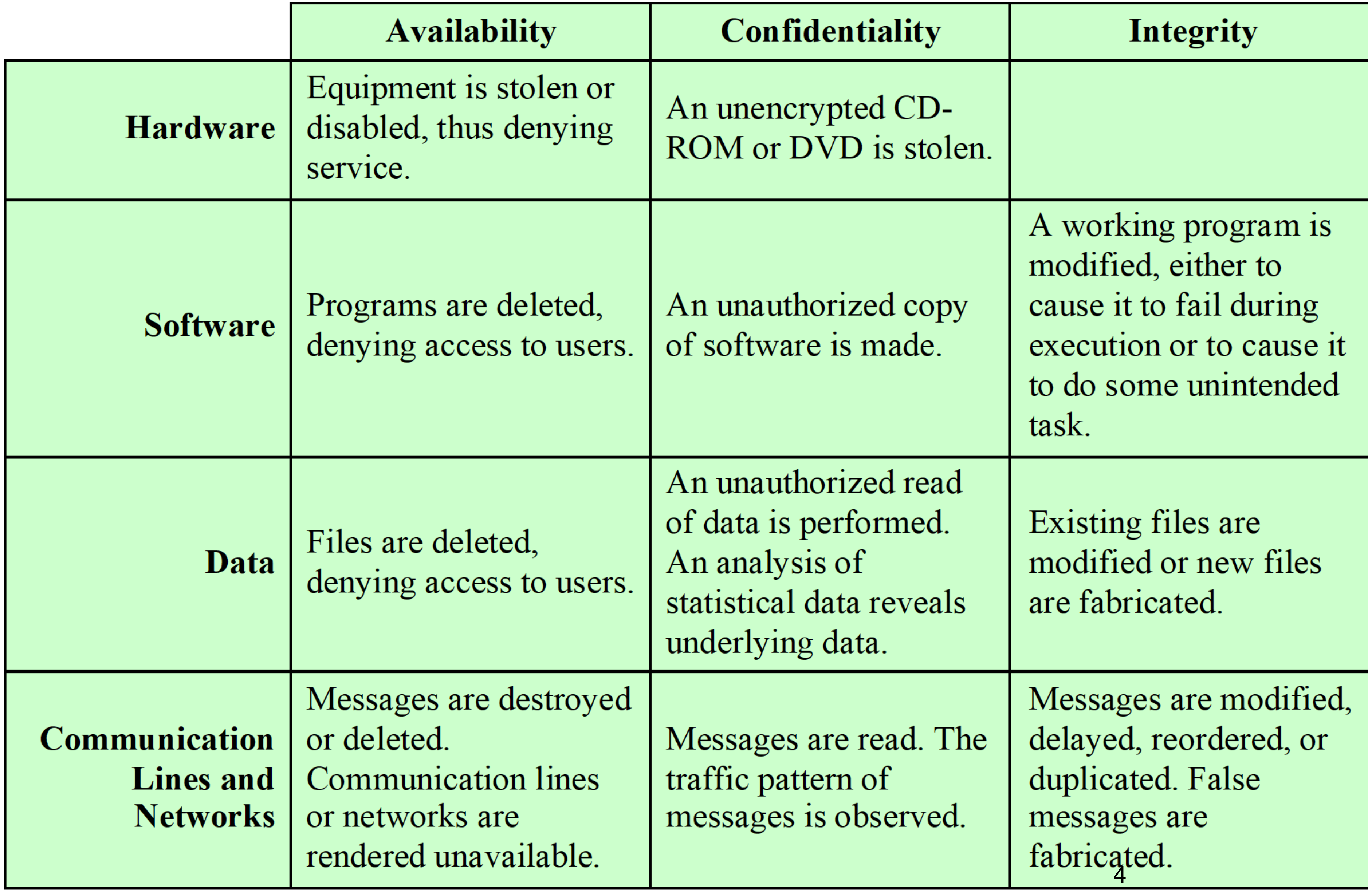
- Overview
- Symmetric encryption
- Public-key cryptography
- Message authentication and hash functions
- Digital signatures
- Random and pseudorandom numbers

The CIA Triad

- Confidentiality: only sender, intended receiver should "understand" message contents
  - sender encrypts message
  - receiver decrypts message
- Integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection; authentication: sender, receiver want to confirm identity of each other
- Availability: services must be accessible and available to users



CIA Examples



	Availability	Confidentiality	Integrity
Hardware	Equipment is stolen or disabled, thus denying service.	An unencrypted CD-ROM or DVD is stolen.	
Software	Programs are deleted, denying access to users.	An unauthorized copy of software is made.	A working program is modified, either to cause it to fail during execution or to cause it to do some unintended task.
Data	Files are deleted, denying access to users.	An unauthorized read of data is performed. An analysis of statistical data reveals underlying data.	Existing files are modified or new files are fabricated.
Communication Lines and Networks	Messages are destroyed or deleted. Communication lines or networks are rendered unavailable.	Messages are read. The traffic pattern of messages is observed.	Messages are modified, delayed, reordered, or duplicated. False messages are fabricated.

Vulnerabilities, Threats and Attacks

- Categories of vulnerabilities
  - Corrupted (loss of integrity)
  - Leaky (loss of confidentiality)
  - Unavailable or very slow (loss of availability)
- Threats
  - Capable of exploiting vulnerabilities
  - Represent potential security harm to an asset
- Attacks (threats carried out)
  - Insider vs. outsider
  - Passive vs. active

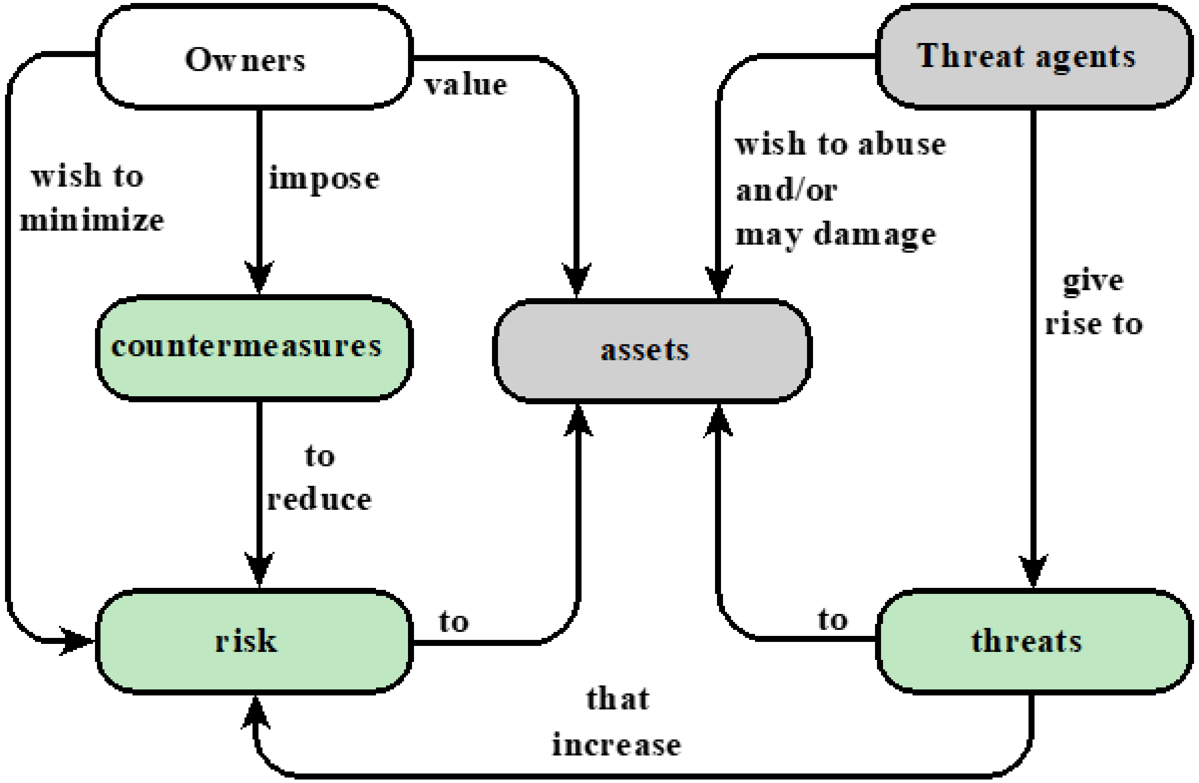


Figure 1.2 Security Concepts and Relationships

Passive Attacks to Compromise Confidentiality

- In passive attacks, attacker attempts to learn or make use of information from the system but does not affect system resources
  - Eavesdropping on, or monitoring of, transmissions
  - Goal of attacker is to obtain information that is being transmitted
- Two categories :
  - Leaking of message content
  - Traffic analysis: attacker infers information from network traffic patterns, even though message content is not leaked (e.g., with encryption)

Active Attacks to Compromise Integrity & Availability

- In active attacks, attacker attempts to alter system resources or affect their operation.
  - Involve some modification of the data stream or the creation of a false stream.
- Four categories:
    - Replay: attacker captures a message and subsequent retransmits it to produce an unauthorized effect.
    - Masquerade / Impersonation: one entity pretends to be a different entity, e.g., can fake (spoof) source address in packet (or any field in packet)
    - Modification of messages: some portion of a message is altered, or messages are delayed or reordered.
    - Denial of Service: prevents or inhibits the normal use of the target system.

Countermeasures

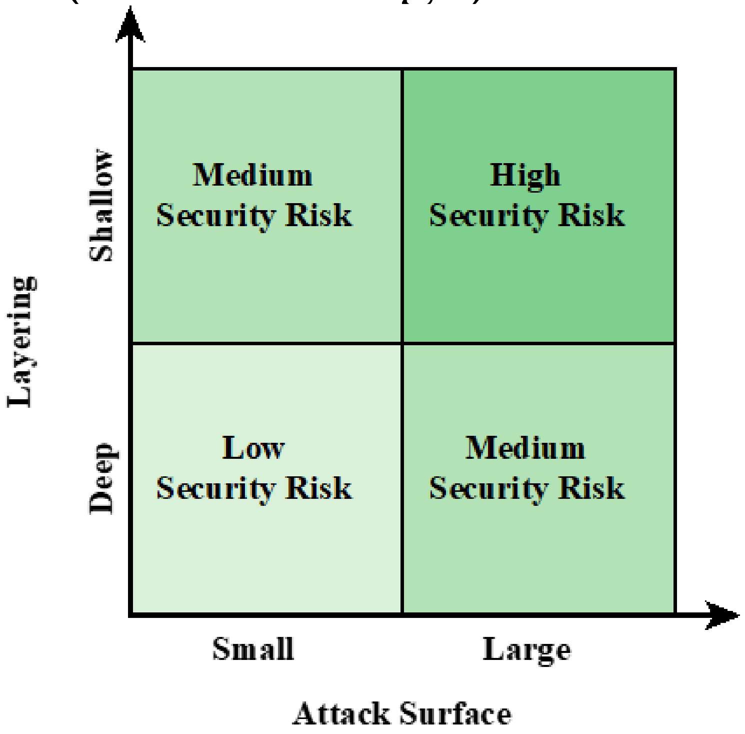
- Means used to deal with security attacks
  - Prevent, Detect, Recover
  - Goal is to minimize residual level of risk to the assets

Attack Surface

- An attack surface consists of the reachable and exploitable vulnerabilities in a system, including:
  - Network Attack Surface
    - Vulnerabilities over an enterprise network, wide-area network, or the Internet
    - Including network protocol vulnerabilities, such as those used for DoS attacks
  - Software Attack Surface
    - Vulnerabilities in application, utility, or operating system code
  - Human Attack Surface
    - Social engineering, human error, and trusted insiders

Security Risk

- Defense in depth is a concept used in information security in which multiple layers of security controls (defense) are placed throughout the system.
- Security risk can be determined by defense in depth layering (shallow or deep) and attack Surface (small or large).



Layering	Small	Large
Shallow	Medium Security Risk	High Security Risk
Deep	Low Security Risk	Medium Security Risk

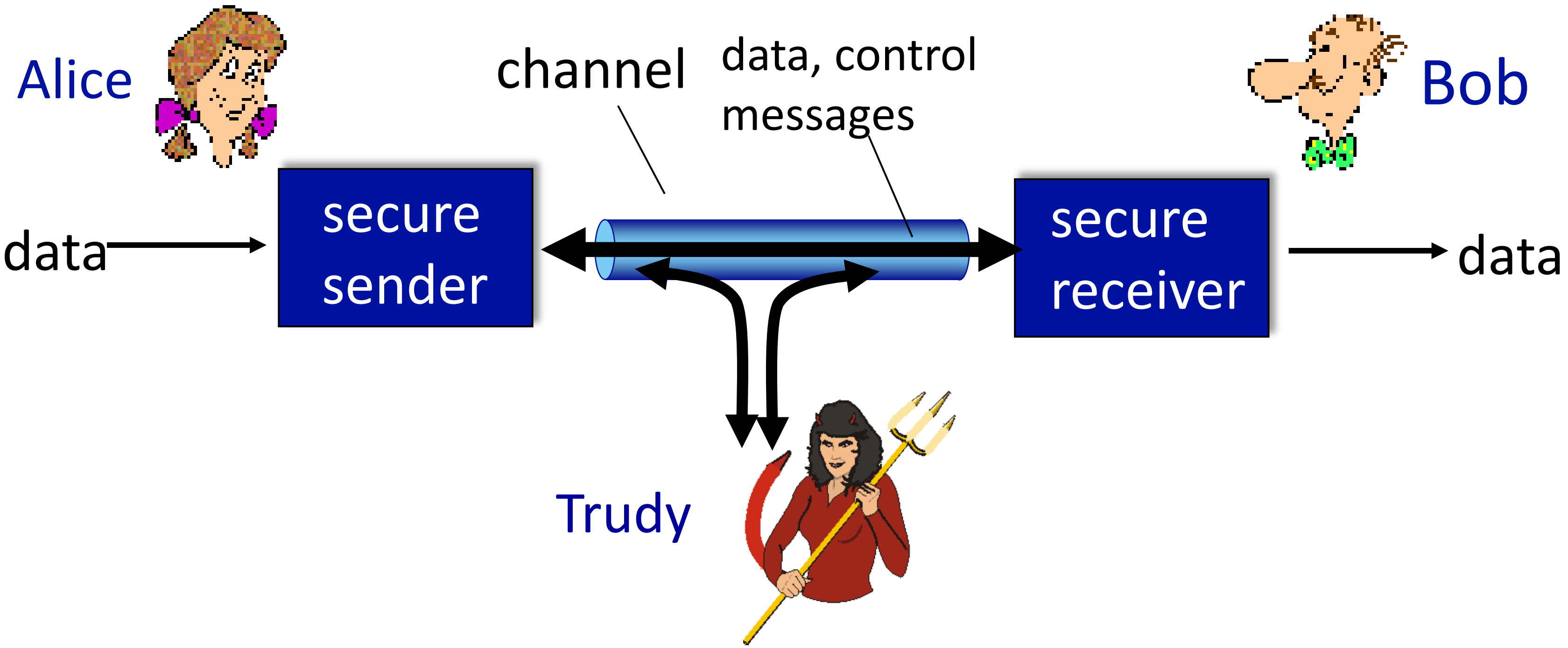
Figure 1.4 Defense in Depth and Attack Surface

Terminology

- Plaintext
  - Also called cleartext
- Ciphertext
  - Scrambled message produced as output
- Encryption algorithm
  - Transforms plaintext to ciphertext
- Decryption algorithm
  - Transforms ciphertext to plaintext

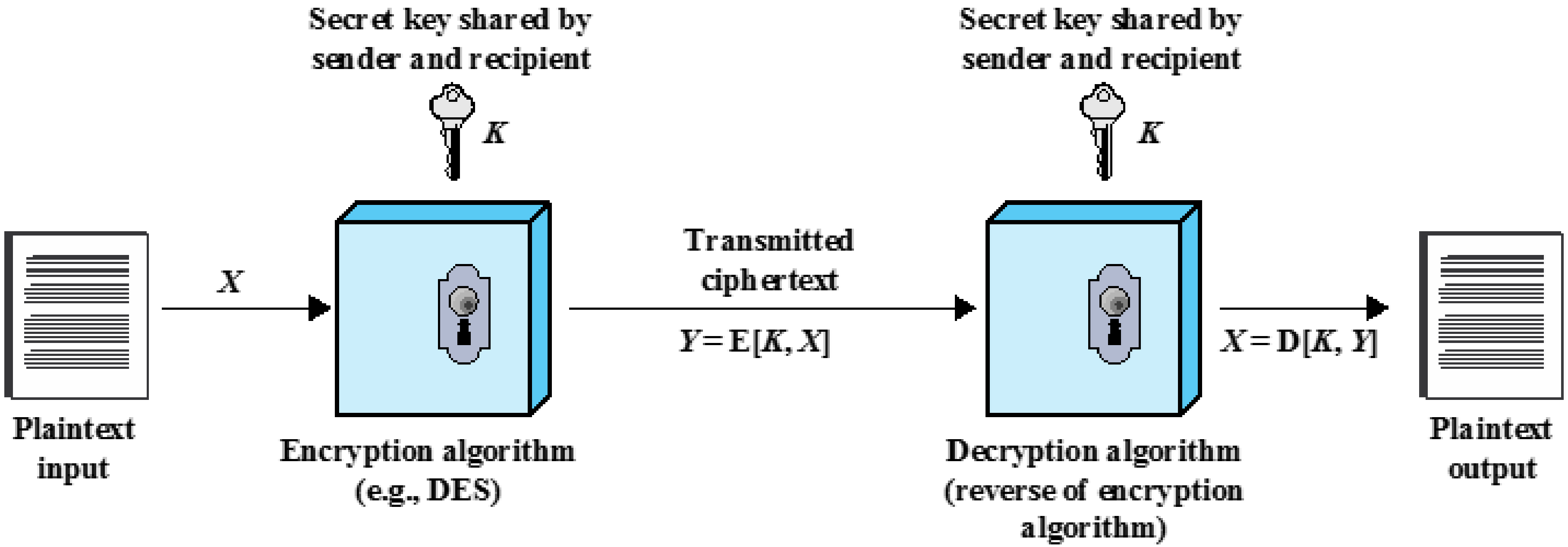
Friends and enemies: Alice, Bob, Trudy

- Well-known in network security world
- Bob, Alice want to communicate “securely”
- Trudy (intruder) may intercept, delete, add messages



Symmetric Encryption

- ❑ Encrypted_Message = Encrypt(Key, Message)
- ❑ Message = Decrypt(Key, Encrypted_Message)
- ▪ Also called secret-key cryptography, for protecting confidentiality
  - Sender and receiver must share the same secret key


Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
Plaintext input
X
Encryption algorithm (e.g., DES)
Transmitted ciphertext
Y = E[K, X]
Decryption algorithm (reverse of encryption algorithm)
X = D[K, Y]
Plaintext output

Simple encryption scheme

substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

```
 plaintext:   abcdefghijklmnopqrstuvwxyz

ciphertext:   mnbvcxzasdfghjklpoiuytrewq
```

e.g.:

```
Plaintext: bob. i love you. alice
ciphertext: nkn. s gktc wky. mgsbc
```

Encryption key: mapping from set of 26 letters to set of 26 letters

A more sophisticated encryption approach

- n substitution ciphers, $M_1, M_2, \ldots, M_n$
- cycling pattern:
  - e.g., n=4: M_1,M_3,M_4,M_3,M_2; M_1,M_3,M_4,M_3,M_2; ..
- for each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
  - dog: d from M_1, o from M_3, g from M_4

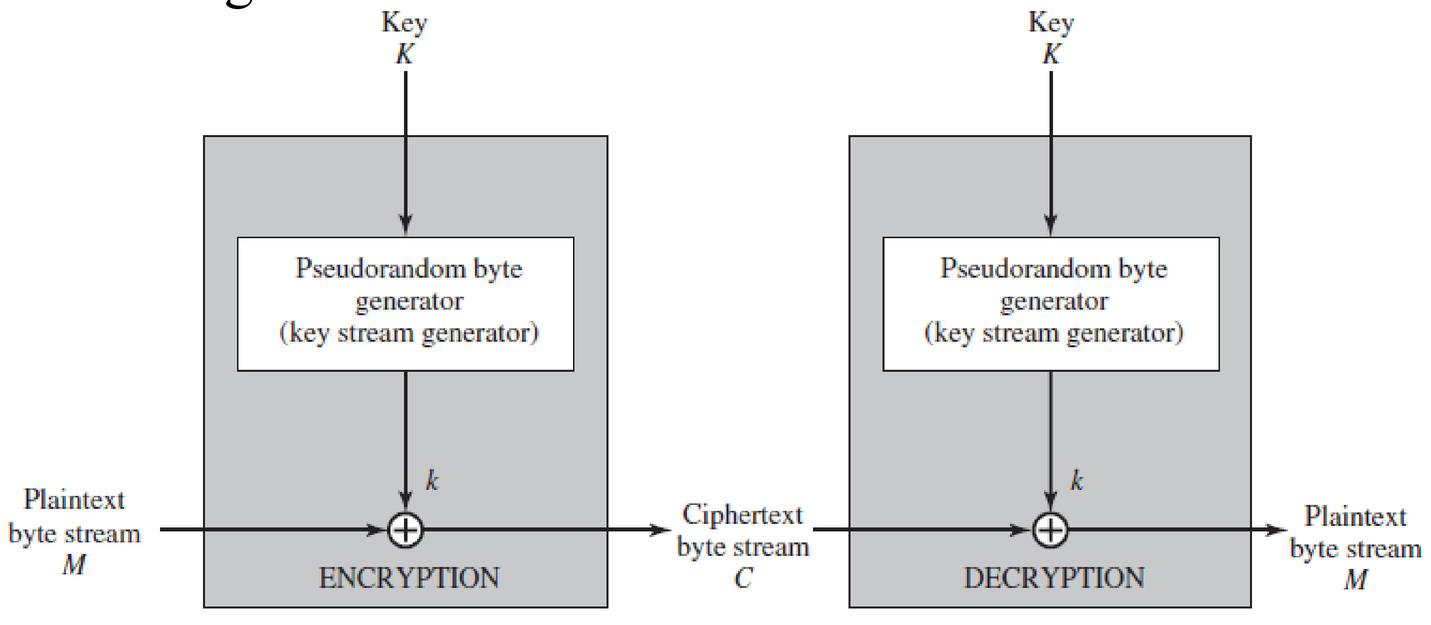
🔑 *Encryption key:* n substitution ciphers, and cyclic pattern

Block & Stream Ciphers

- Stream Cipher
  - Processes the input elements continuously, producing output one element at a time
  - One element may be 1 bit, 1 Byte, or more than 1 Byte
  - Faster than block ciphers
- Block Cipher
  - Processes input data one block at a time
  - Produces an output block for each input block
  - We focus on block cipher in this lecture

An Example Stream Cypher

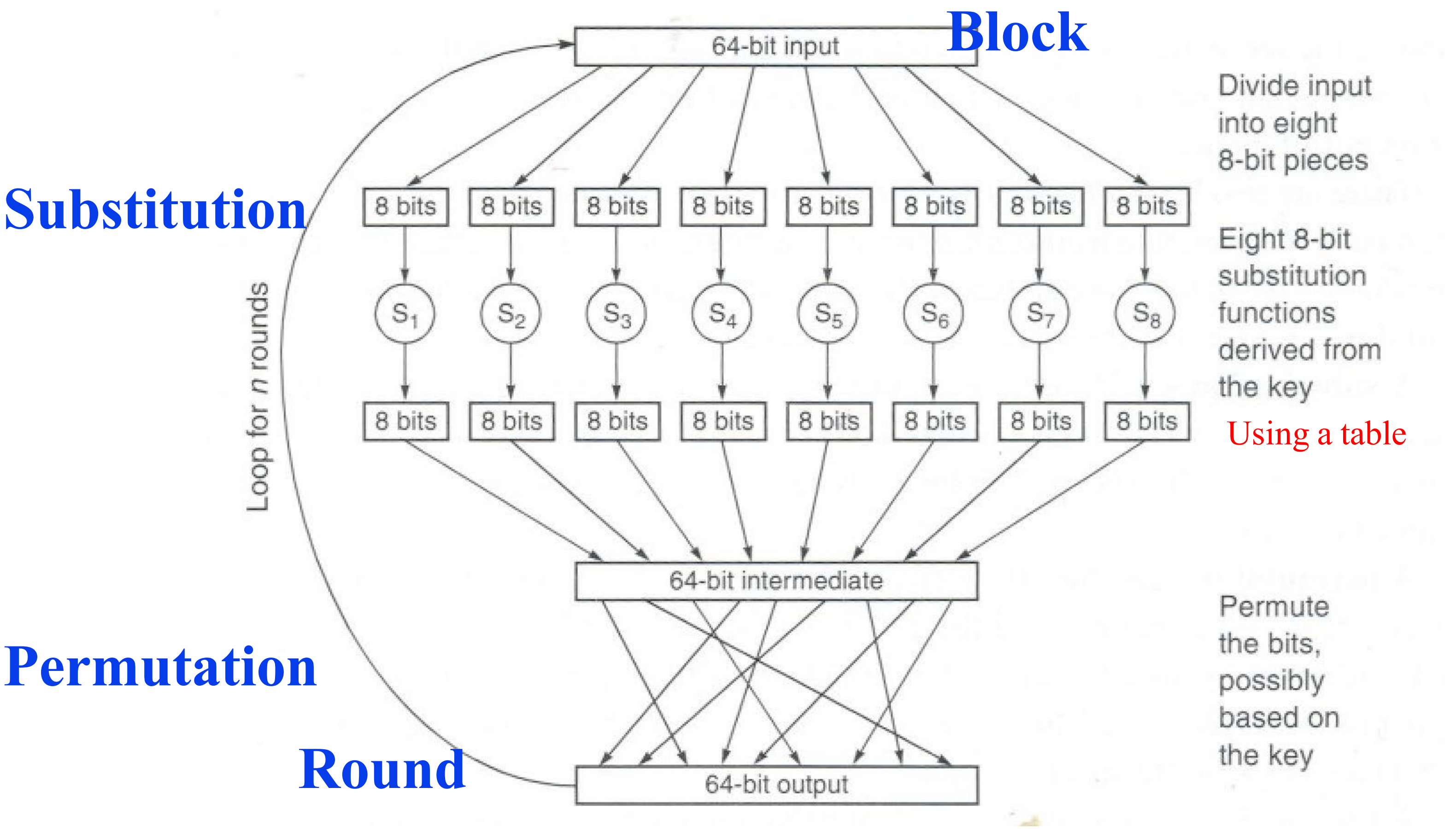
- A stream cypher that stream cipher that operates one bit at a time:
  - Sender and receiver share a secrete key K, which can be input to a pseudorandom byte generator that produces a pseudorandom stream of bytes, called a keystream KS(k in the figure).
  - The plaintext is XOR'ed with KS bit-by-bit to produce the cyphertext: $C_i = M_i\ XOR\ KS_i$
  - The cyphertext is XOR'ed with the same keystream KS to recover the plaintext $M_i = C_i\ XOR\ KS_i$
  - This relies on sender and receiver sharing a secrete key K and using the same key stream generator algorithm


Key
K
Pseudorandom byte generator (key stream generator)
k
Plaintext byte stream M
ENCRYPTION
Ciphertext byte stream C
Key
K
Pseudorandom byte generator (key stream generator)
k
DECRYPTION
Plaintext byte stream M

(b) Stream encryption

Block Encryption

- Block Encryption for each block

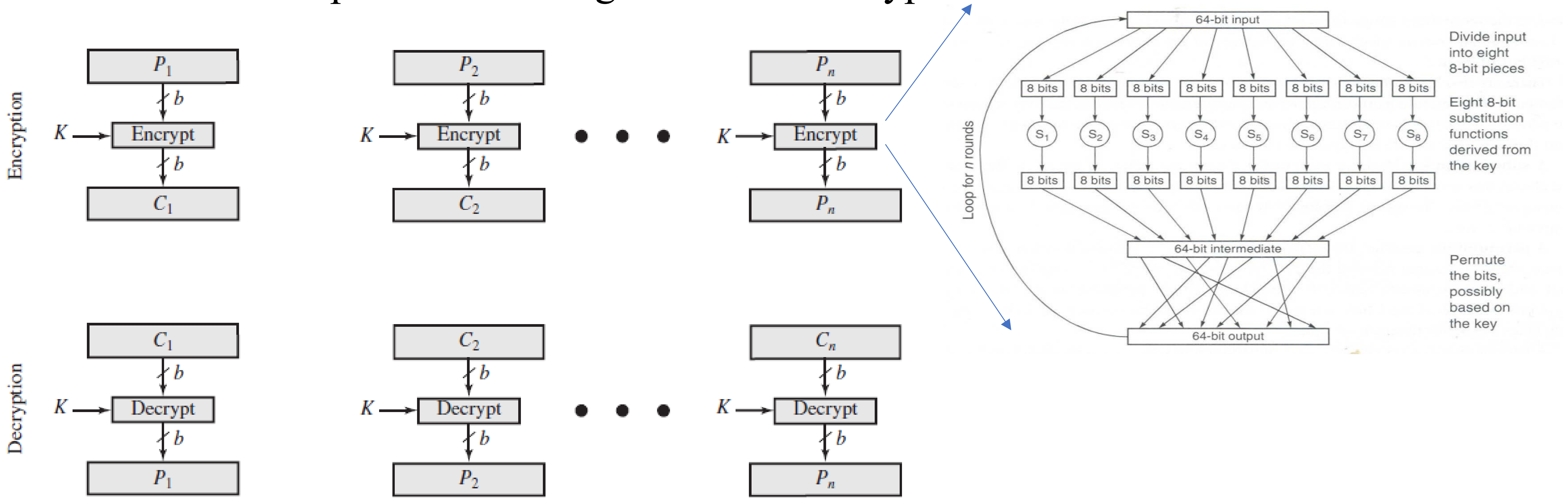


Block Encryption (Cont)

- 64-bit block
  - Short block length ⇒ tabular attack
- Transformations:
  - Substitution: replace k-bit input blocks with k-bit output blocks
  - Permutation: move input bits around. 1 → 13, 2 → 61, etc.
- Round: Substitution round followed by permutation round to achieve Diffusion + Confusion.
  Diffusion ⇒ 1-bit change in input changes many bits in the output.
  Confusion ⇒ Relationship between input and output is complex.

Block Cipher: Electronic CodeBook (ECB)

- Each plaintext block P_i (e.g., 64 or 128 bits) is encoded independently using the same key K
  - Attacker may exploit regularities in the plaintext to perform cryptoanalysis, since same plaintext block generates same cyphertext block.



(a) Block cipher encryption (electronic codebook mode)

Block Cipher: Cipher Block Chaining (CBC)

- Goal: The same block is encoded differently each time.
- Input to the encryption algorithm is the XOR of the plaintext block P_i and the preceding ciphertext block C_{i-1}. The first block is XORed with an Initialization Vector (IV),
  - P_i and C_{i-1} have no fixed relationship, removing the regularities in the plaintext for ECB, hence more secure.
- Exclusive OR (XOR) operator
  - 0 XOR 0 = 0
  - 0 XOR 1 = 1
  - 1 XOR 0 = 1
  - 1 XOR 1 = 0
  - 0101 XOR 0011 = 0110
  - Properties
    - $C_{j-1}\ XOR\ C_{j-1} = 0$ (any string XOR'ed with itself is all 0's).
    - $0\ XOR\ P_j = P_j$ (0 XOR'ed with any string is the same string).

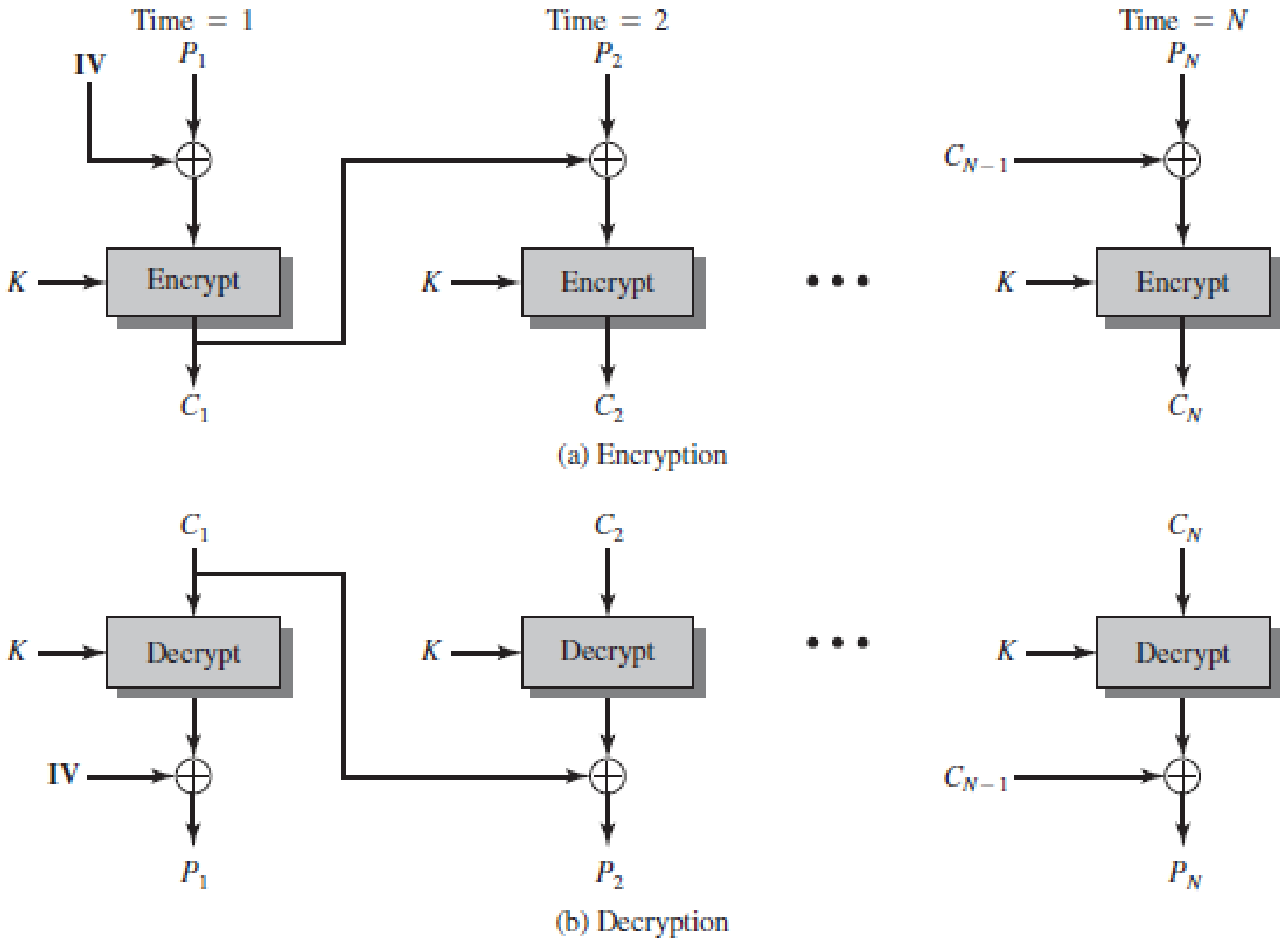


Figure 20.7 **Cipher Block Chaining (CBC) Mode**

Data Encryption Standard (DES)

- Published by National Institute of Standards and Technology (NIST) in 1977
- For commercial and *unclassified* government applications
- Eight-octet (64-bit) key.
- Efficient hardware implementation
- Used in most financial transactions
- Computing power goes up one bit every two years
- 56-bit was secure in 1977 but is not secure today

Triple DES (3DES)

- Repeats DES three times using either 2 or 3 unique keys, with total key length of 112 or 168 bits
  - Ciphertext= DES(K1, DES(K2, DES(K1, Plain Text))) or
  - Ciphertext= DES(K3, DES(K2, DES(K1, Plain Text)))
- Pros:
  - Key length of 112/168 bits removes the vulnerability to brute-force attacks
- Cons:
  - Performance is slow: three times as many calculations as DES.
  - Block size larger than 64 bits is desirable for efficiency and security.

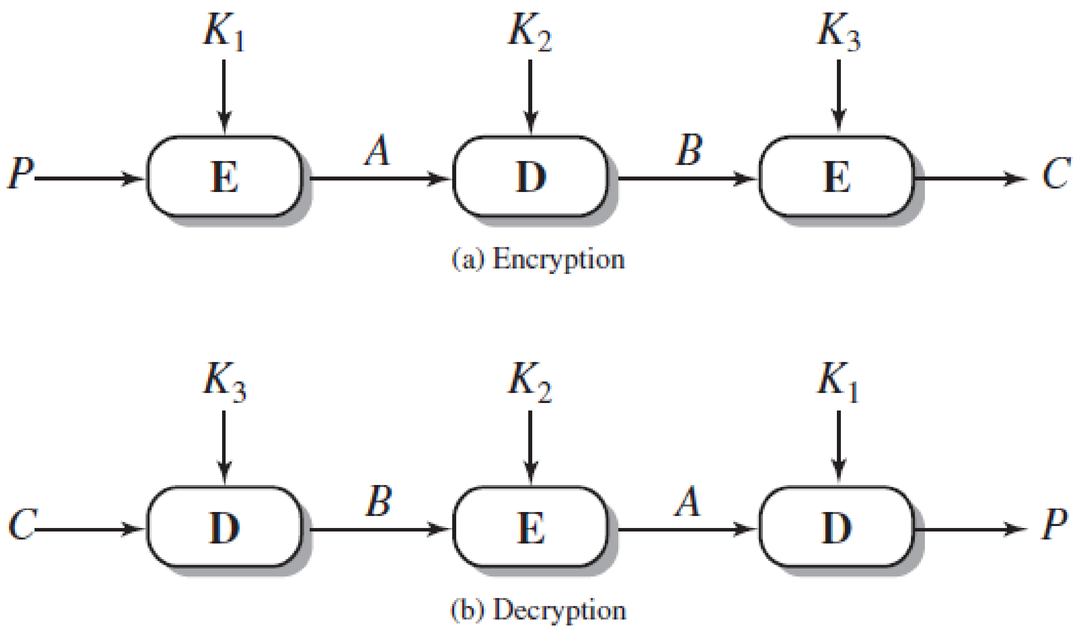
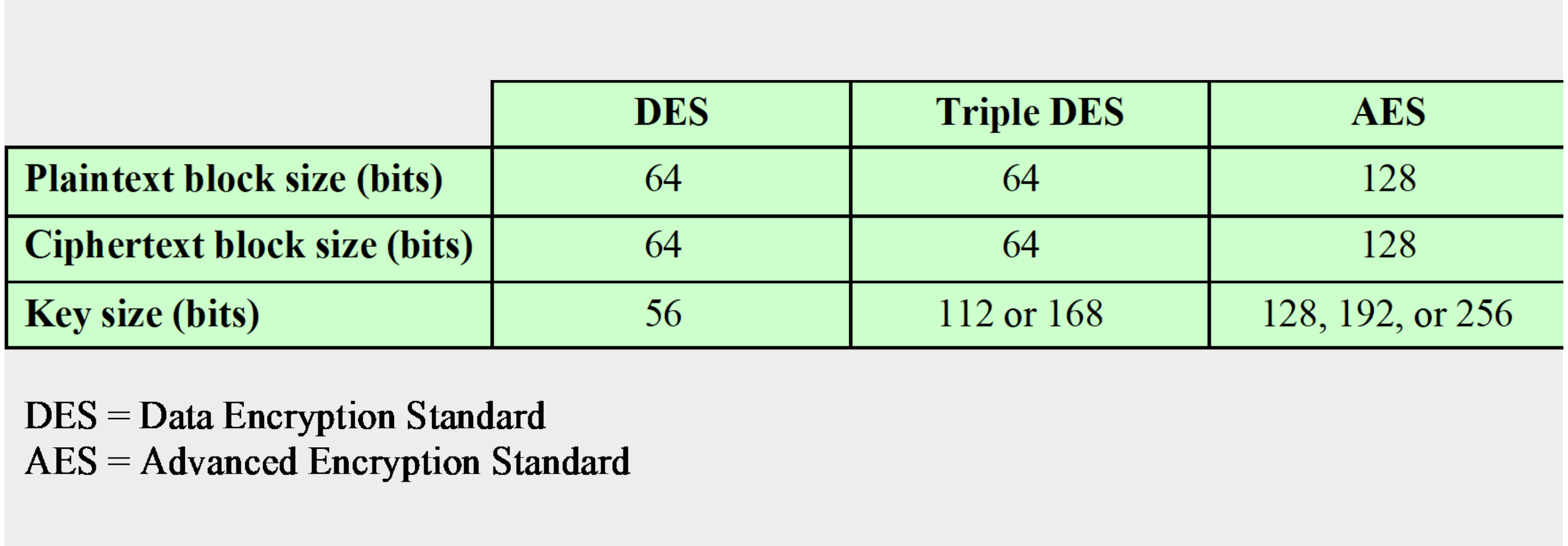


Figure 20.2 **Triple DES**

Advanced Encryption Standard (AES)

- Designed in 1997-2001 by National Institute of Standards and Technology (NIST) as Federal information processing standard (FIPS 197)
- A symmetric block cipher with a block length of 128 bits
- Key lengths 128, 192, and 256 bits.



	DES	Triple DES	AES
Plaintext block size (bits)	64	64	128
Ciphertext block size (bits)	64	64	128
Key size (bits)	56	112 or 168	128, 192, or 256

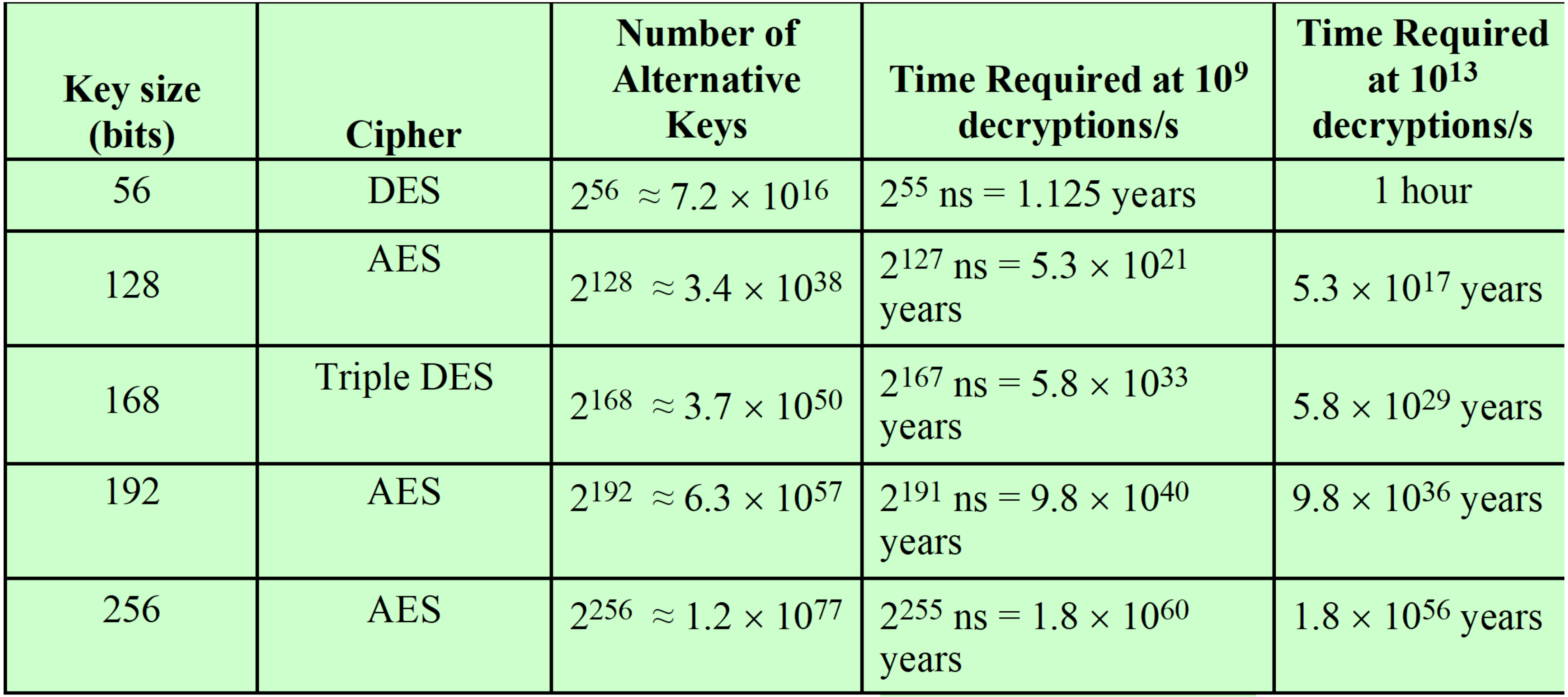
DES = Data Encryption Standard
AES = Advanced Encryption Standard

Attacks against Symmetric Encryption

- Cipher-text only attack: Trudy has ciphertext she can analyze
- Two approaches:
  - brute force: search through all keys
  - statistical analysis

- Known-plaintext attack: Trudy has plaintext corresponding to ciphertext
  - *e.g.,* in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,
- Chosen-plaintext attack: Trudy can get ciphertext for chosen plaintext

Time Required for Brute-Force Attack



Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions/s	Time Required at 10^{13} decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years

Symmetric Key Encryption: Summary

1. Secret key encryption requires a shared secret key
2. Block encryption, e.g., DES, 3DES, AES, break cleartext into fixed-size blocks and encrypt
3. CBC ensures that the same plain text results in different ciphertexts.

Quiz

- [6 points] Consider the 3-bit block cipher in the Table below

Plain	000	001	010	011	100	101	110	111
Cipher	110	111	101	100	011	010	000	001

- Suppose the plaintext is 100101100.

(a) Initially assume that CBC is not used. What is the resulting ciphertext?

(b) Suppose Trudy sniffs the ciphertext. Assuming she knows that a 3-bit block cipher without CBC is being employed (but doesn’t know the specific cipher), what can she surmise?

(c) Now, suppose that CBC is used with IV-111. What is the resulting ciphertext?

Quiz ANS

Plain	000	001	010	011	100	101	110	111
Cipher	110	111	101	100	011	010	000	001

(a) Initially assume that CBC is not used. What is the resulting ciphertext?

ANS: Ciphertext for plaintext 100101100 is 011010011, since 100 maps to 011, 101 maps to 010, 100 maps to 011

(b) Suppose Trudy sniffs the ciphertext. Assuming she knows that a 3-bit block cipher without CBC is being employed (but doesn't know the specific cipher), what can she surmise?

ANS: Without CBC, each identical plaintext block will always map to the same ciphertext block. This makes it easier for Trudy to recognize patterns in repeated blocks of data. If Trudy intercepts enough ciphertexts, she could perform frequency analysis on the blocks. For example, if certain ciphertext blocks appear more frequently, she might guess that they correspond to more common plaintext blocks (like spaces or common letters in text). Or if it is known that the message always starts out with certain predefined fields, then the cryptanalyst may have a number of known plaintext-ciphertext pairs to work with.

(c) Now, suppose that CBC is used with IV=111. What is the resulting ciphertext?

ANS: With CBC and IV = 111, resulting ciphertext for plaintext 100101100 is 100111100. See next page.

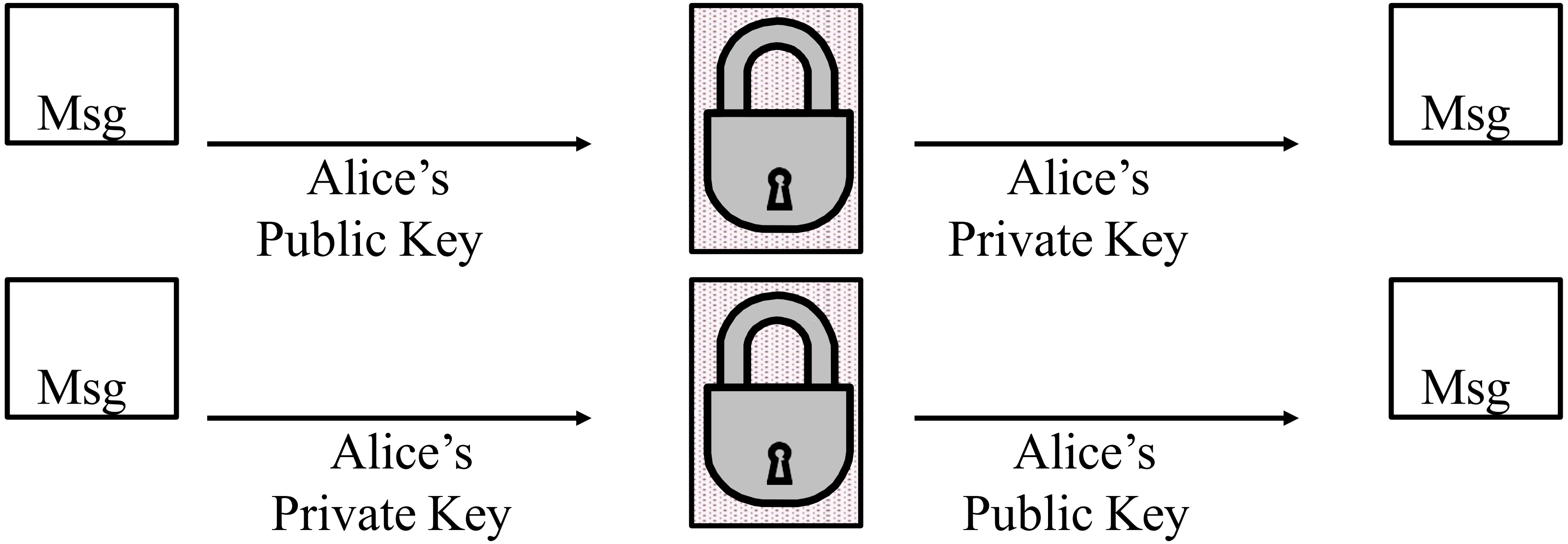
Quiz ANS cont

Plain	000	001	010	011	100	101	110	111
Cipher	110	111	101	100	011	010	000	001

- The first step is to XOR the first plaintext block with IV = 111
  - First plaintext block: 100, so 100⊕111=011
  - Now we encrypt this result (011) using our cipher table: 011 maps to 100.
- Second Block: Now we XOR the second plaintext block with the first ciphertext block:
  - Second plaintext block: 101, so 101⊕100=001
  - Now we encrypt this result (001) using our cipher table: 001 maps to 111.
- Third Block: Finally, we XOR the third plaintext block with the second ciphertext block:
  - Third plaintext block: 100, so 100⊕111=011
  - Now we encrypt this result (011) using our cipher table: 011 maps to 100.
- Resulting ciphertext for plaintext 100101100 is 100111100.

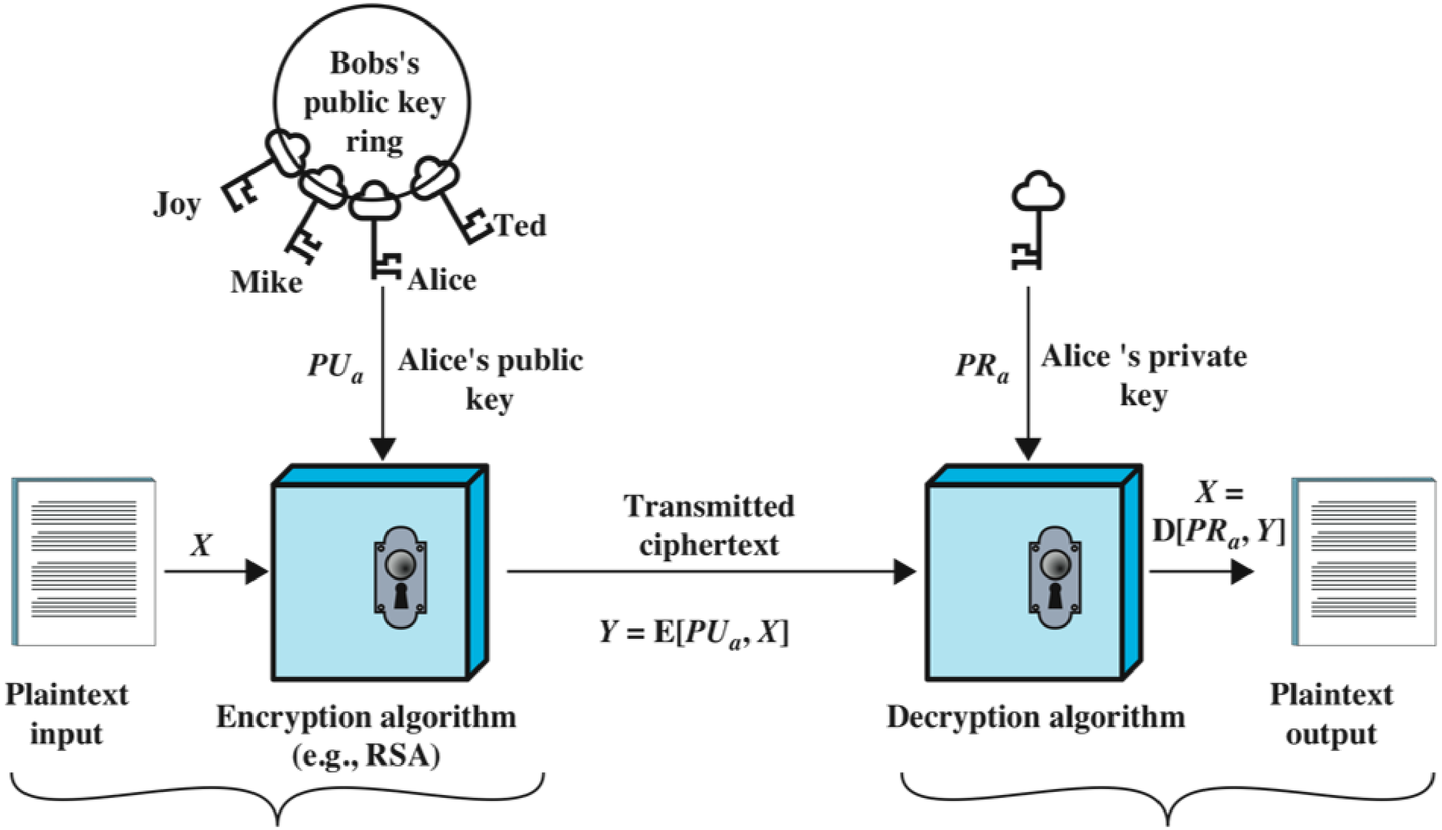
Public Key Cryptography

- Each user has two separate keys
  - PU_a: Alice’s Public Key; PR_a: Alice’s Private Key
  - PU_b: Bob’s Public Key; PR_b: Bob’s Private Key
- Encrypted with private key. decrypted with public key
  - Message=Decrypt(PU_a, Encrypt(PR_a, Message))
- Encrypted with public key, decrypted with private key
  - Message=Decrypt(PR_a, Encrypt(PU_a, Message))
- Requirement: given public key, it is computationally infeasible to compute private key



Public-Key Crypto for Confidentiality

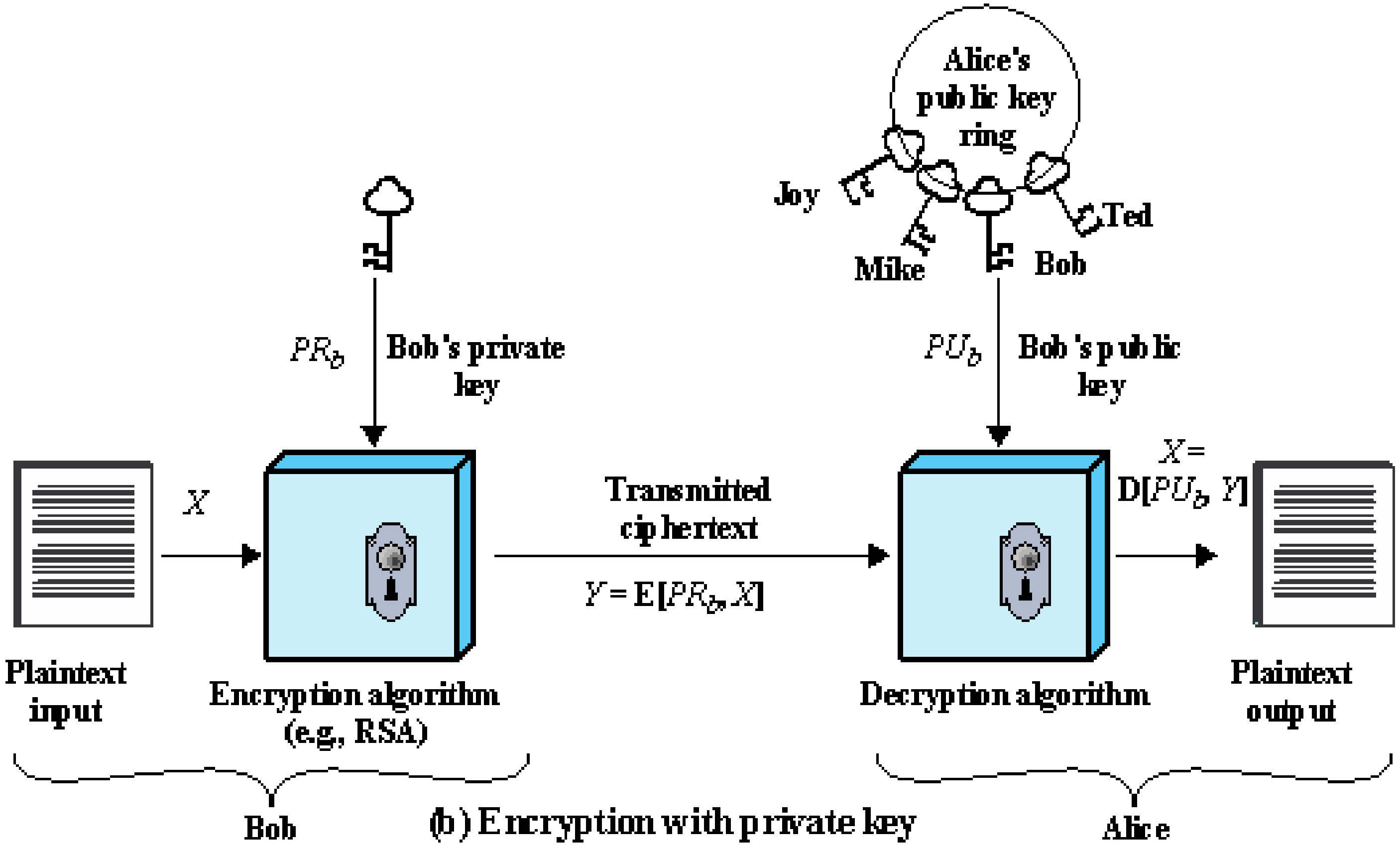
- Sender encrypts data using the receiver’s public key
- Receiver decrypts data using his own private key
  - Use of a public/private key pair removes the need for sharing a secret key



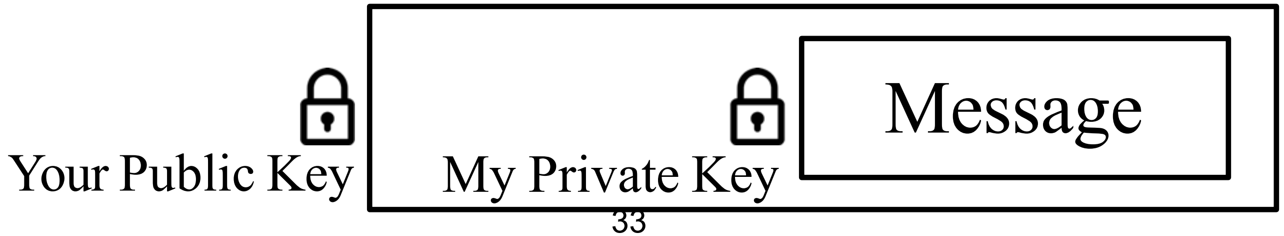
(a) Encryption with public key

Public-Key Crypto for Integrity and Non-Repudiation

- Sender encrypts data using his or her private key
- Receiver, or anyone else, can decrypt the message using sender's public key.
  - There are more efficient methods based on MAC or crypto hash function (discussed later).

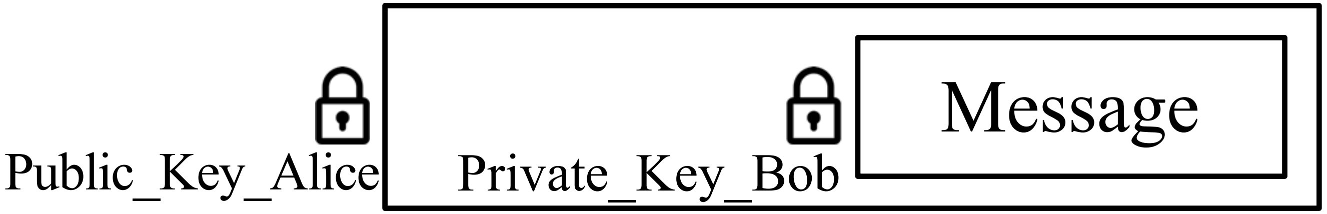


(b) Encryption with private key



Public-Key Crypto for both Confidentiality and Integrity/Non-Repudiation

- Combine the previous two approaches
- Bob sends message to Alice
- Bob performs: Encrypted_Message = Encrypt(PU_a ublic_Key_Alice, Encrypt(PR_b, Message))
- Alice performs: Cleartext_Message = Decrypt(Public_Key_Bob, Decrypt(Private_Key_Alice, Encrypted_Message)



Public-Key Crypto Algorithms and Protocols

- RSA (Rivest-Shamir-Adelman)
  - Key generation, encryption, decryption
- Elliptic Curve Cryptography (ECC)
  - Lightweight public key algorithm for embedded and IoT devices
- Diffie-Hellman
  - Key exchange protocol used to establish a shared secret between two parties, e.g., a secret key for symmetric encryption

Prerequisite: Modular Arithmetic

- x mod n = remainder of x when divide by n
- facts:

  [(a mod n) + (b mod n)] mod n = (a+b) mod n

  [(a mod n) - (b mod n)] mod n = (a-b) mod n

  [(a mod n) * (b mod n)] mod n = (a*b) mod n
- thus

  $(a \text{ mod } n)^d \text{ mod } n = a^d \text{ mod } n$
- example: x=14, n=10, d=2:

  $(x \text{ mod } n)^d \text{ mod } n = 4^2 \text{ mod } 10 = 6$

  $x^d = 14^2 = 196 \quad x^d \text{ mod } 10 = 6$

Diffie-Hellman

- Alice and Bob agree on:
  - A sufficiently large prime number p
  - A base number $g < p$
- Alice chooses a secret number a and sends to Bob
  - $A = g^a \bmod p$
- Bob chooses a secret number b and sends to Alice
  - $B = g^b \bmod p$
- Alice computes $S_1 = B^a \bmod p$
- Bob also computes $S_2 = A^b \bmod p$
- We can easily show that $S_1 = S_2$, and this is their shared secret key
  - $S_1 = B^a \bmod p = g^{ab} \bmod p = A^b \bmod p = S_2$
- p, g, A, B are all public; only Alice knows secret a; only Bob knows secret b. At the end, Alice and Bob have a shared secret $S_1 = S_2$.

Requirements for Public-Key Crypto

- Computationally easy to
  - create key pairs
  - encrypt/decrypt messages using either public or private key
- Computationally infeasible to
  - determine private key from public key
  - recover cleartext without key
- Either key can be used for each role (public/private key)

Public-Key Crypto in Practice

- Public-key crypto, e.g., RSA is computationally intensive
  - DES is at least 100 times faster than RSA
- In practice: use public key crypto to establish secure connection, then use symmetric encryption for encrypting data

session key, K_S

- Bob and Alice use public-key crypto to exchange a symmetric session key K_S
- Then use K_S for symmetric encryption/decryption

Public Key Crypto: Summary

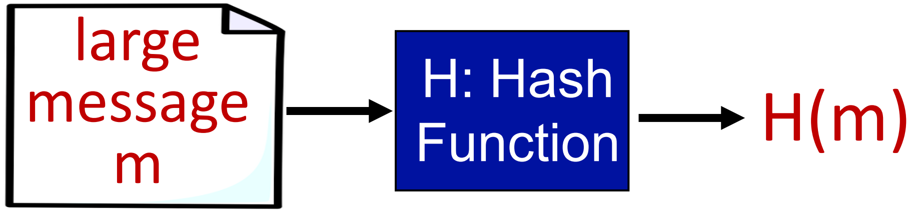
1. Public Key Encryption uses two keys: Public and Private.
2. Either key can be used to encrypt. The other key will decrypt.

Message Authentication

- Verifies received message is authentic, i.e., Contents unchanged + Source Verified
  - Its content has not been altered (authentic content), and it is from the alleged sender (authentic source)
- Can use public-key crypto
  - Sender Bob encrypts the message with his private key, receiver Alice decrypts it with Bob’s public key
  - But computationally inefficient for large messages

Message Authentication Approaches

- Message Authentication Code (MAC)
  - Sender and receiver have shared secret key K.
  - Computes $MAC_M = F(K, M)$ for input message M, with secret key K, as a fixed-length output (e.g., 16 or 32 bits), e.g., using symmetric encryption algorithm AES to generate MAC of 128 bits.
- Cryptography Hash Function
  - Does not require a shared secret key K.
  - Computes a hash $H(M)$ for input message M of any size, with fixed-length output (e.g., 128-512 bits), also called a message digest.



Message Authentication Code (MAC)

- Sender and receiver share a secret key K. Sender calculates MAC as a function of message and key: $MAC_M = F(K,M)$.
- Receiver recomputes the MAC with $F(K, receivedM)$, and compares it with the received MAC_M. If they match, then message is authenticated: $receivedM = M$, and it is from the alleged sender with secrete key K.

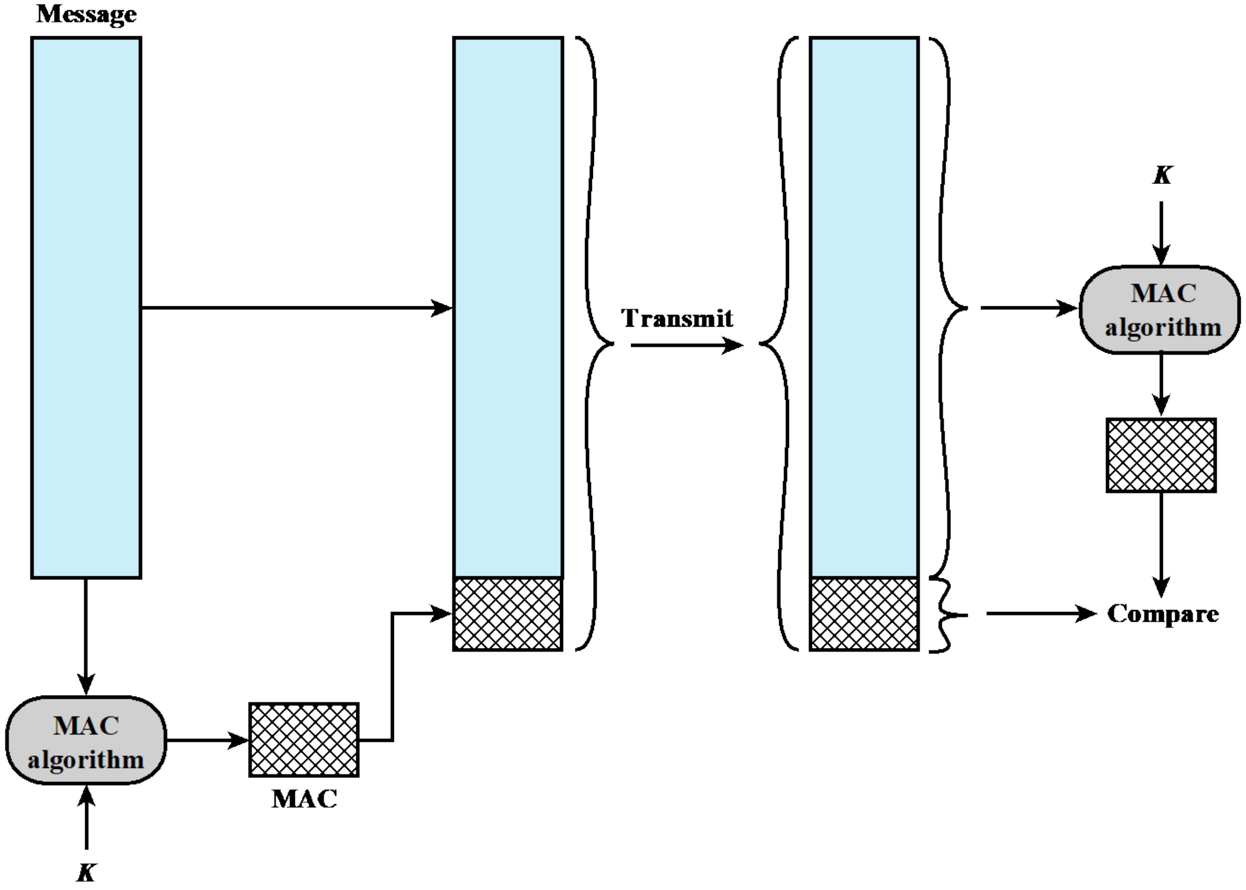
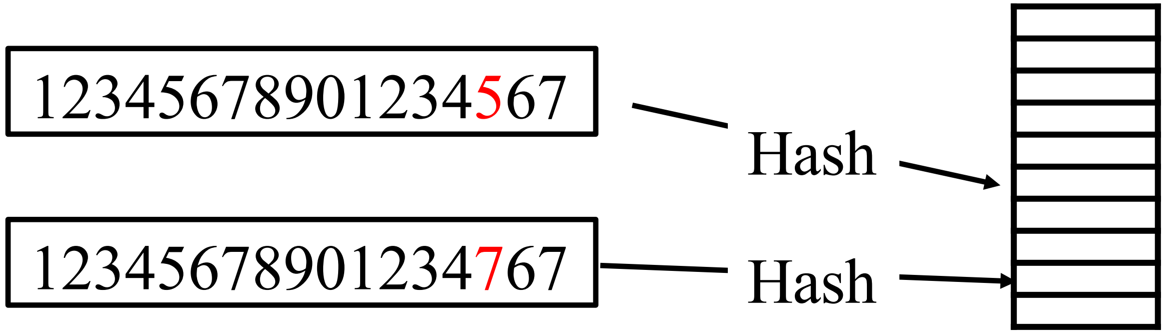


Figure 2.3 Message Authentication Using a Message Authentication Code (MAC).

MAC Explanations

- If only the sender and receiver know the secret key K, and if the received MAC matches the receiver-computed MAC, then
- 1. The receiver is assured that the message has not been altered.
  - If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker does not know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.
- 2. The receiver is assured that the message is from the alleged sender.
  - Because no one else knows the secret key K, no one else could prepare a message with a proper MAC.
- 3. If the message includes a sequence number (such as in TCP)
  - Then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

Hash Functions



12345678901234567

12345678901234767

Hash

Hash

Example: CRC can be used as a hash
(not recommended for security applications)

Requirements:

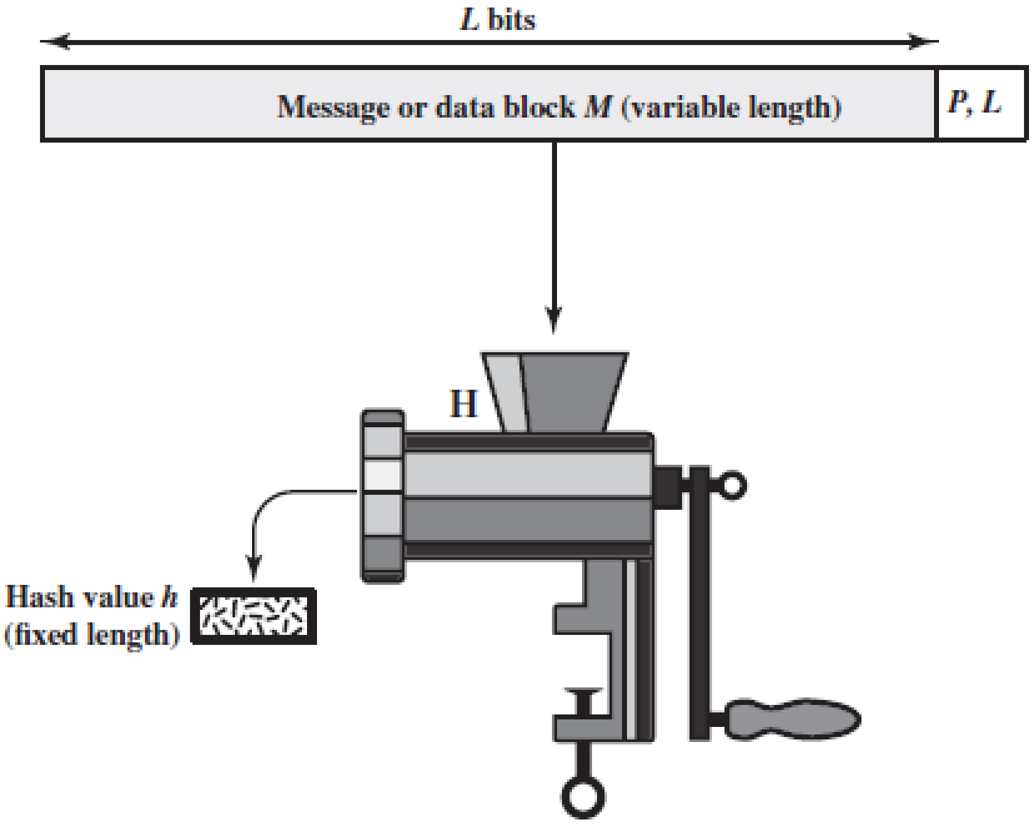
1. Applicable to any size message
2. Fixed length output
3. Easy to compute
4. Difficult to Invert ⇒ Can’t find *x* given H(*x*) ⇒ One-way
5. Difficult to find y, such that H(*x*) = H(*y*) ⇒ Can’t change msg
6. Difficult to find *any* pair (*x*, *y*) such that H(*x*) = H(*y*)
   ⇒ Strong hash

Hash Function Applications

- UNIX password checking
  - password hash values are stored in the file /etc/passwd. When user tries to log in, system computes hash of user-entered password and compares with stored password hash
- Intrusion detection
  - Hash value $H(F)$ for a file F is stored in a secure location to detect any alteration of file contents

Crypto Hash Function

- The crypto hash function accepts a variable-size message M as input and produces as output a fixed-size message digest, $H(M)$.
- Given message digest x, computationally infeasible to find message M such that $x = H(M)$, e.g.,
  - ❑ MD5
    - ❑ 128-bit hash using 512-bit blocks using 32-bit operations
  - ❑ SHA-1 Algorithm
    - ❑ 160-bit hash using 512-bit blocks and 32-bit operations



P, *L* = padding plus length field

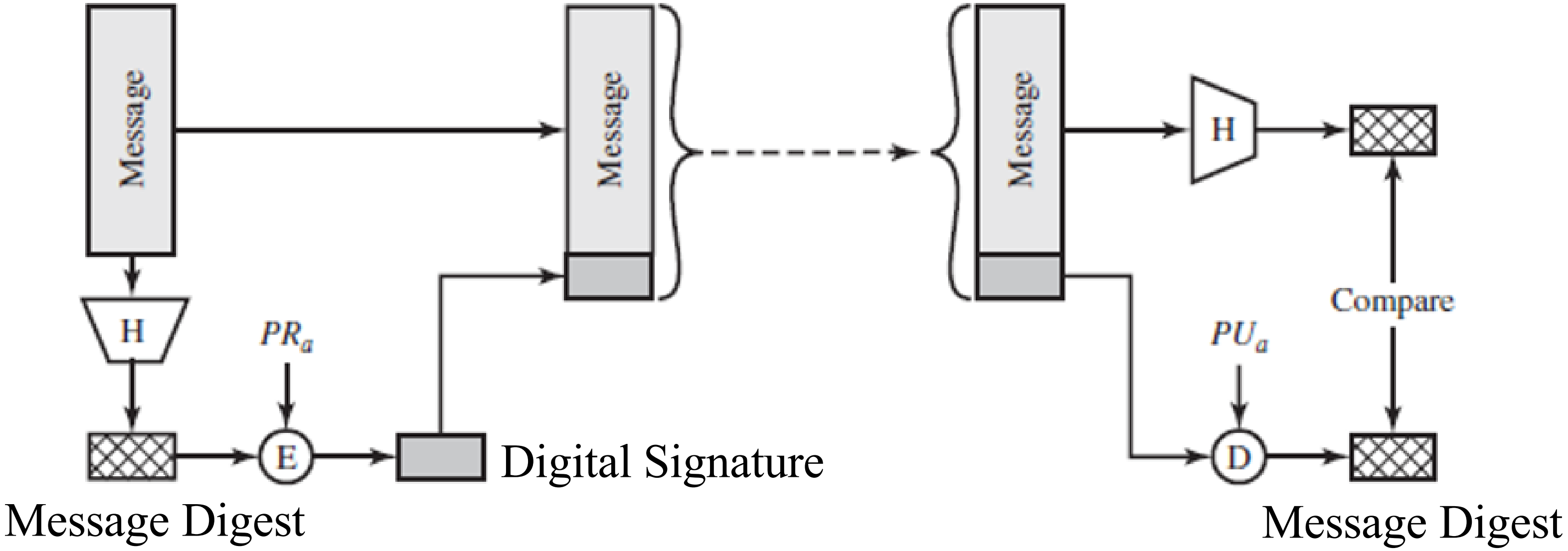
Figure 2.4 Cryptographic Hash Function; $h = H(M)$

Crypto Hash Function Requirements

- One-way function: Easy to compute $H(x)$ given x, but computationally infeasible to find x given $H(x)$.
- Weak collision resistance: given x, computationally infeasible to find $y \neq x$ such that $H(y) = H(x)$
  - Otherwise, attacker can substitute a fake message y for a given authentic message x
- Strong collision resistance: computationally infeasible to find any pair $(x, y), x \neq y$, such that $H(x) = H(y)$
  - Otherwise, attacker Bob can generate two messages x and y with same $H(x) = H(y)$. Msg x is an IOU for $10, and Msg y is an IOU for $100. Bob sends x to Alice, who computes $H(x)$ and encrypts it with her private key as signature of message x. Bob can use Msg y in conjunction with $H(x)$ as proof that Alice owes him $100 instead of $10.

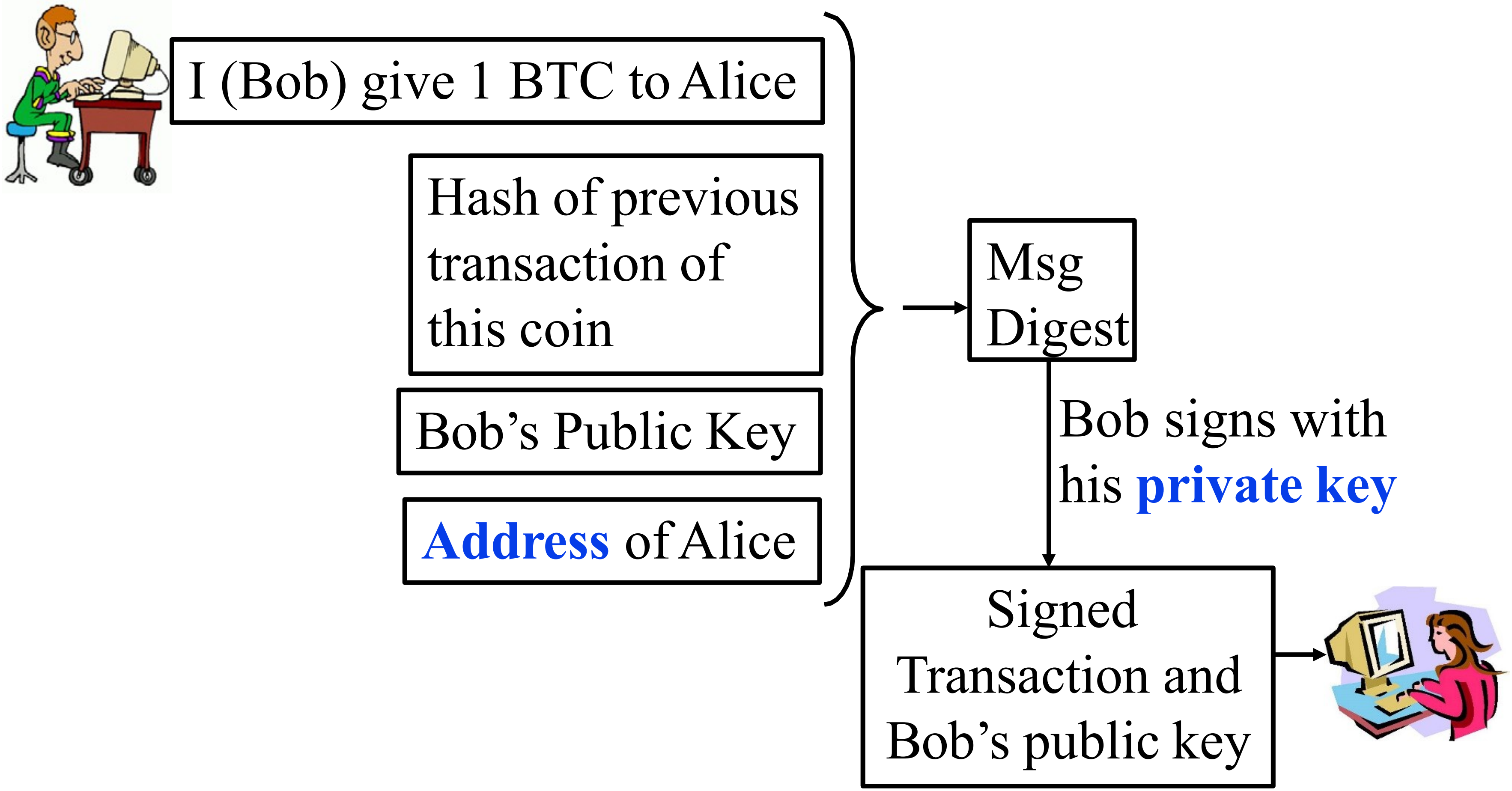
Digital signature

- A digital signature is a cryptographic technique used to verify the authenticity and integrity of a digital message or document. It serves a similar purpose to a handwritten signature or a stamped seal, but it provides far more security due to its cryptographic foundations.
- Apply crypto hash function to generate message digest $H(M)$, which is encrypted with sender's private key PR_a to generate a digital signature. Receiver decrypts $H(M)$ with sender's public key PU_a. and compares it to the recomputed message digest $H(receivedM)$. If they match, then message is authenticated.
  - More efficient than MAC approach, since encryption/decryption is applied to the message digest, not the entire message



Application of Digital signature: Bitcoin

- A transaction: “Bob gives 1 BTC to Alice"



I (Bob) give 1 BTC to Alice

Hash of previous transaction of this coin

Bob’s Public Key

Address of Alice

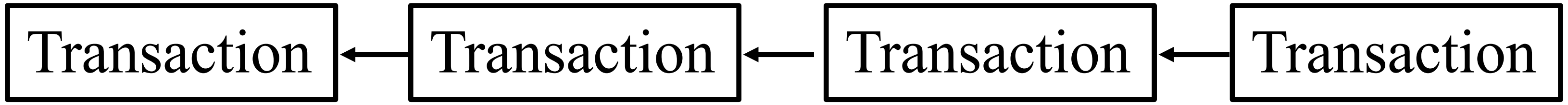
Msg Digest

Bob signs with his **private key**

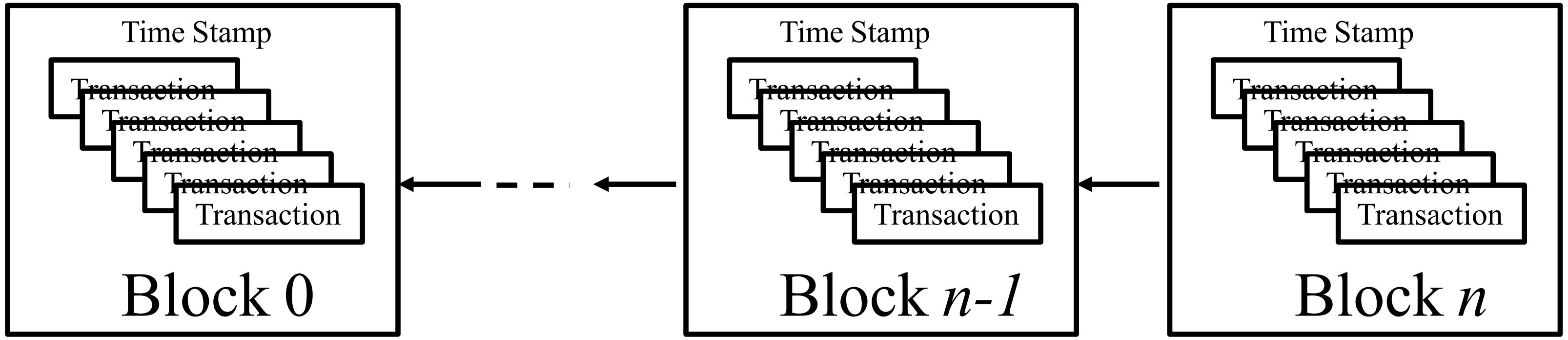
Signed Transaction and Bob’s public key

Blockchain

- Transaction Chain:

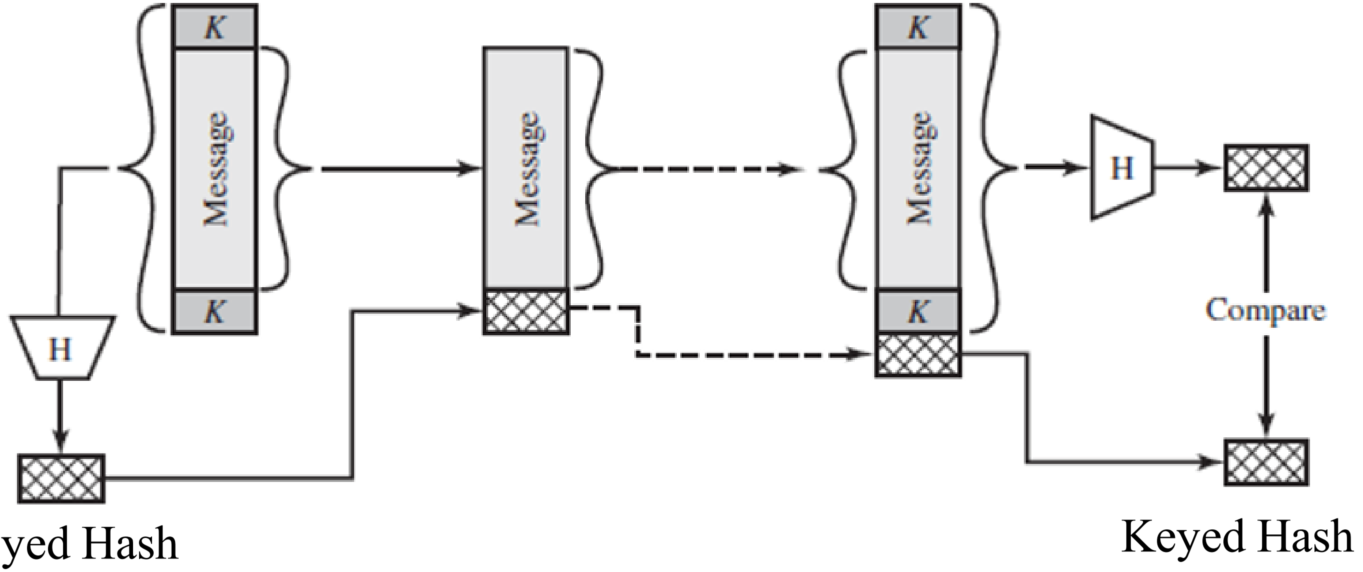


- Problem:
  - Too many transactions ⇒ Chain too long
  - Takes too long to find and verify a transaction
- Solution: Combine several transactions into blocks of verified transactions



HMAC

- HMAC (Hash-based Message Authentication Code): sender and receiver share a secret key K. Apply hash function $H()$ on the concatenation of secret key K and the message to generate Keyed Hash $H\left(K||M||K\right)$. Receiver recomputes $H(K||receivedM||K)$ and compares it with received Keyed Hash.
  - HMAC is a general design. Can use any hash function ⇒ HMAC-MD5, HMAC-AES.
  - Efficient, since it does not perform encryption/decryption.
- HMAC can ensure both the integrity and authenticity of a message. It combines a cryptographic hash function with a secret key to generate a keyed hash that can only be generated or verified by someone who knows the secret key.

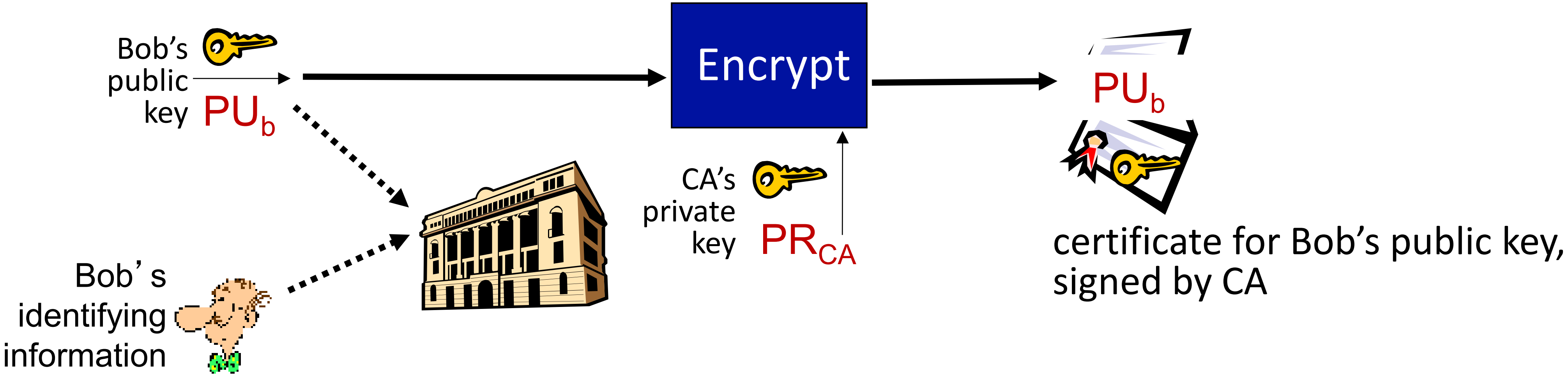


Need for Certified Public Keys

- Motivation: Trudy plays pizza prank on Bob
  - Trudy creates e-mail order:
    Dear Pizza Store, Please deliver to me four pepperoni pizzas. Thank you, Bob
  - Trudy signs order with her private key
  - Trudy sends order to Pizza Store
  - Trudy sends to Pizza Store her public key, but says it's Bob's public key
  - Pizza Store verifies signature; then delivers four pepperoni pizzas to Bob

Public key Certification Authority (CA)

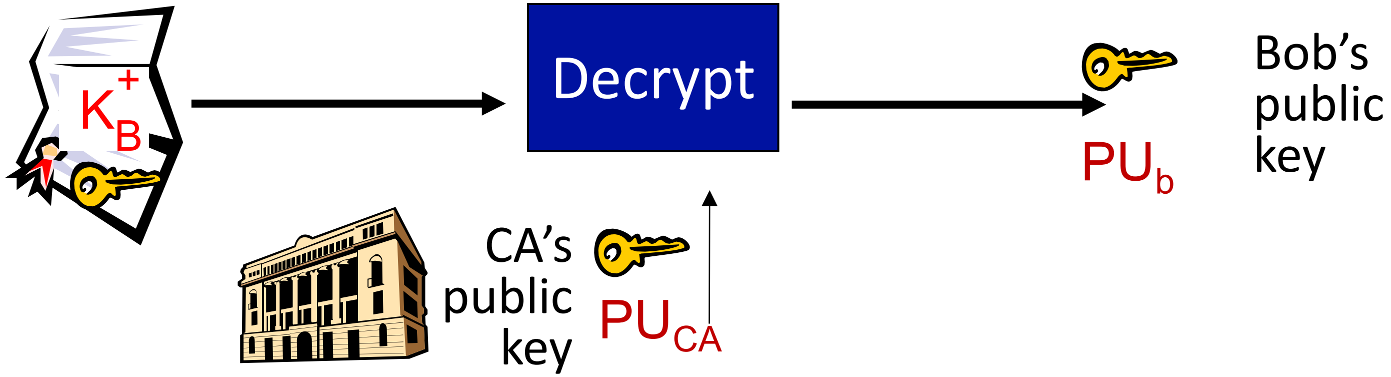
- Certification Authority (CA) binds public key to particular entity, E
- Entity Bob (person, website, router) registers its public key with CE provides "proof of identity" to CA
  - CA creates certificate binding identity Bob to his public key
  - certificate containing Bob's public key digitally signed by CA: CA says "this is Bob's public key"



certificate for Bob's public key, signed by CA

Public key Certification Authority (CA)

- When Alice wants Bob’s public key:
  - Gets Bob’s certificate (Bob or elsewhere)
  - Apply CA’s public key to Bob's certificate, get Bob’s public key



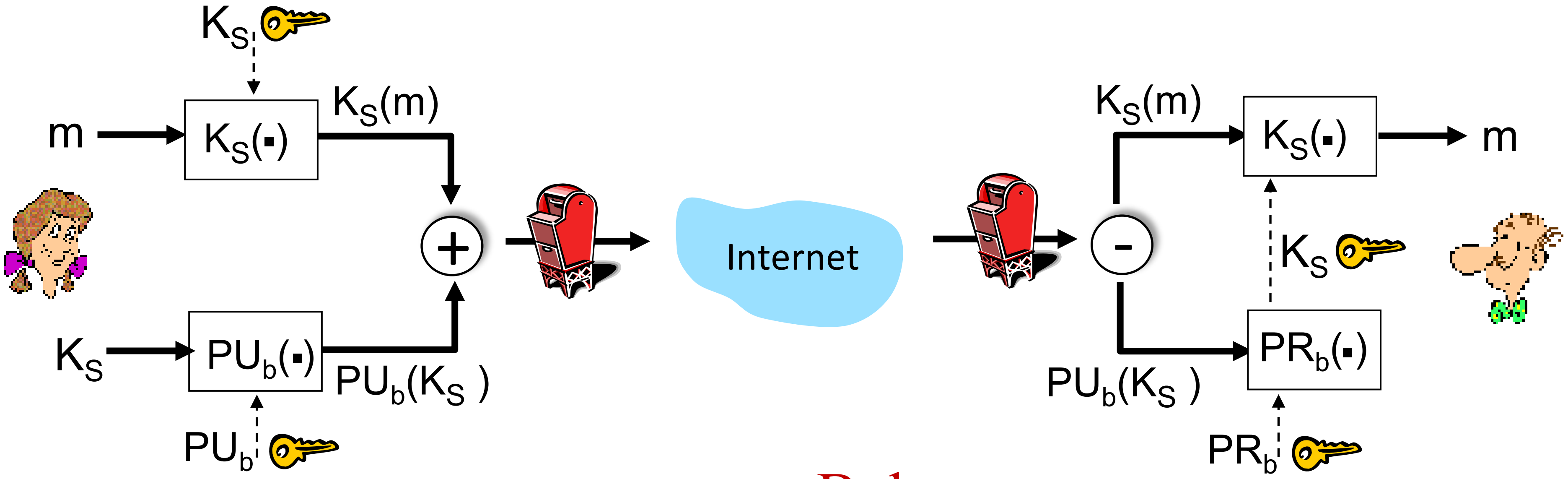
Hashes, Signatures, Certificates: Summary

1. Hashes are one-way functions such that it is difficult to find another input with the same hash, like MD5 and SHA-1.
2. Message Authentication Code (MAC) ensures message integrity and source authentication using hash functions.
3. Digital Signature consists of encrypting the hash of a message using the private key.
4. Digital certificates are signed by Certification Authorities (CAs) and contain public keys.

Secure e-mail: confidentiality

Q: Why not encrypt the whole email with Bob's public key?
A: Since public key crypto is inefficient, this would be too slow
Recall [Slide 39 "Public-Key Crypto in Practice"]

Alice wants to send *confidential* e-mail, m, to Bob.



Alice:

- generates random secrete key, K_S
  - Called a session key, since it is used for this session only
- encrypts message with K_S (for efficiency)
- also encrypts K_S with Bob's public key
- sends both $K_S(m)$ and $PU_b(K_S)$ to Bob

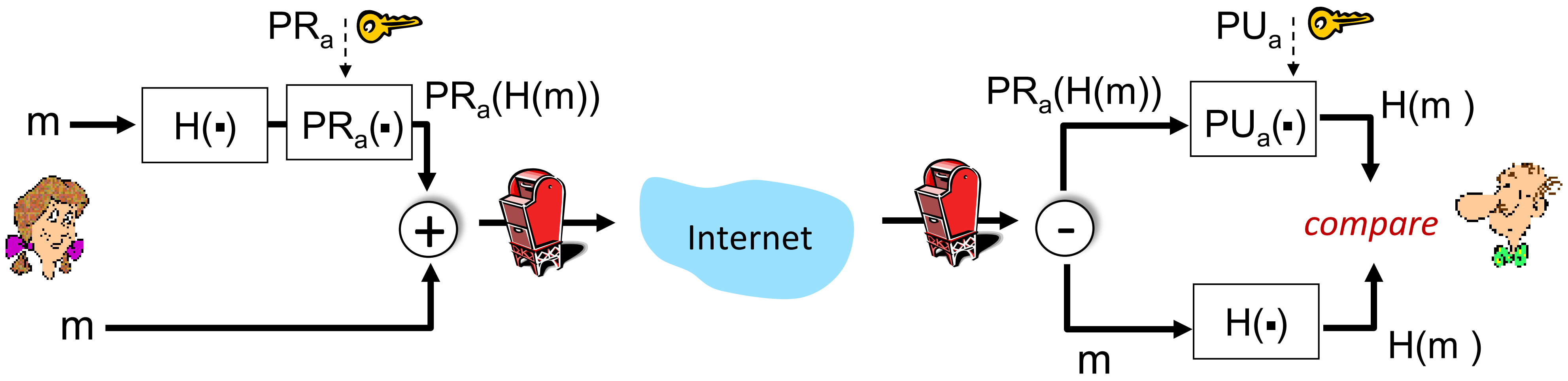
Bob:

- uses his private key to decrypt and recover K_S
- uses K_S to decrypt $K_S(m)$ to recover m

Secure e-mail: integrity, authentication

Recall Slide 49 “Digital Signature”

Alice wants to send m to Bob, with *message integrity*, *authentication*



Alice:

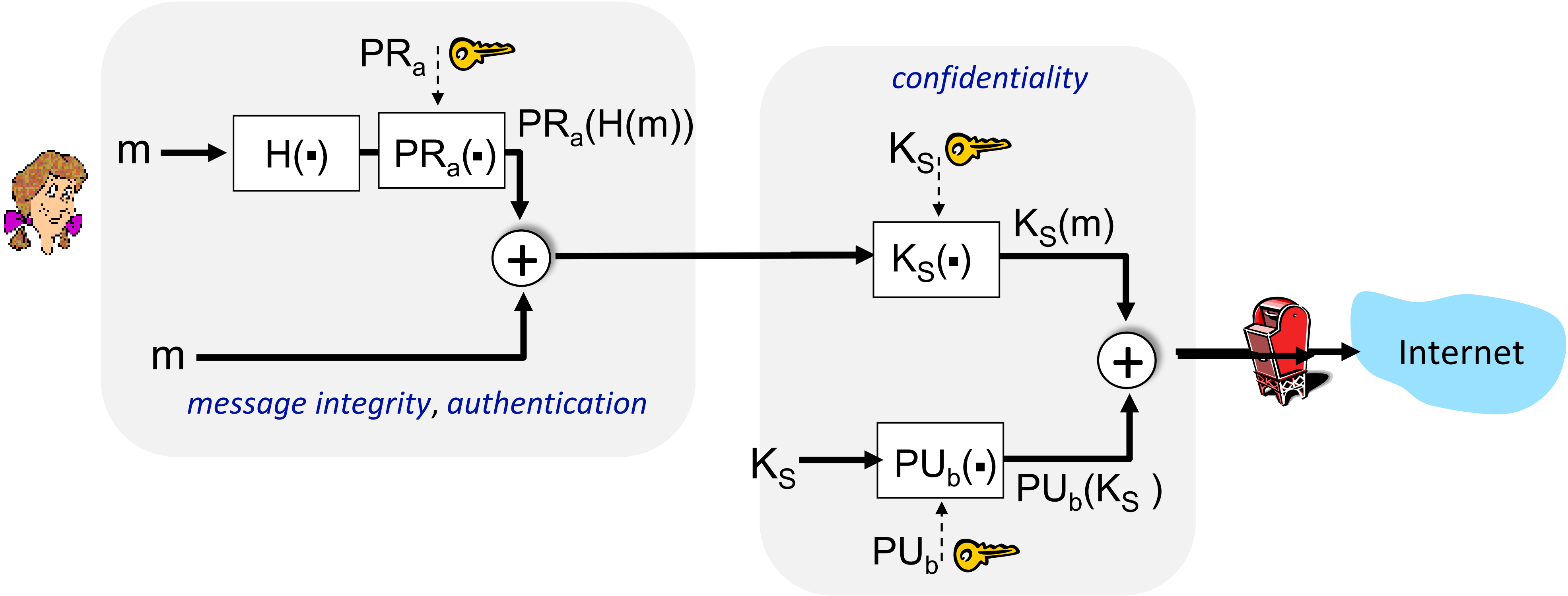
- Digitally signs message digest $H(m)$ with her private key to create a digital signature
- sends both message (cleartext) and digital signature to Bob
  - Does not protect confidentiality

Bob:

- uses Alice’s private key to decrypt and recover $H(m)$
- Compares with the recomputed $H(m)$

Secure e-mail: confidentiality, integrity, authentication

Alice sends m to Bob, with *confidentiality, message integrity*, *authentication*



Alice uses three keys: her private key PR$_a$, Bob’s public key PU$_b$, session key K$_S$

What are Bob’s complementary actions?

Random Numbers

- Used to generate:
  - Keys for public-key algorithms
  - Stream key for symmetric stream cipher
  - Symmetric key for use as a temporary session key or in creating a digital envelope
  - Handshaking to prevent replay attacks
  - Session key

Random Number Requirements

- Randomness
  - Uniform distribution
    - Frequency of occurrence of each of the numbers should be approximately the same
  - Independence
    - No one value in the sequence can be inferred from the others
- Unpredictability
  - Each number is statistically independent of others in the sequence, so future elements of the sequence cannot be predicted based on past elements

Pseudorandom vs. Random Numbers

- Pseudorandom numbers are generated with deterministic algorithms with random seeds:
  - The same seed results in the same sequence of random numbers
  - The sequence produced are not truly statistically random, but may pass many reasonable tests of randomness
- True random number generator (TRNG):
  - Uses a nondeterministic source to produce randomness
  - Most operate by measuring unpredictable natural processes
    - e.g. radiation, gas discharge, leaky capacitors

Summary

1. Network security requires confidentiality, integrity, availability, authentication, and non-repudiation.
2. Encryption can use one secret key or two keys (public and private)
3. Public key crypto is very compute-intensive and is generally used to send the secret key
4. A digital certificate system is used to certify the public key
5. Secure e-mail uses confidentiality using a secret key, uses certificates and public keys to sign the e-mail and send the secret key