ECE 4309

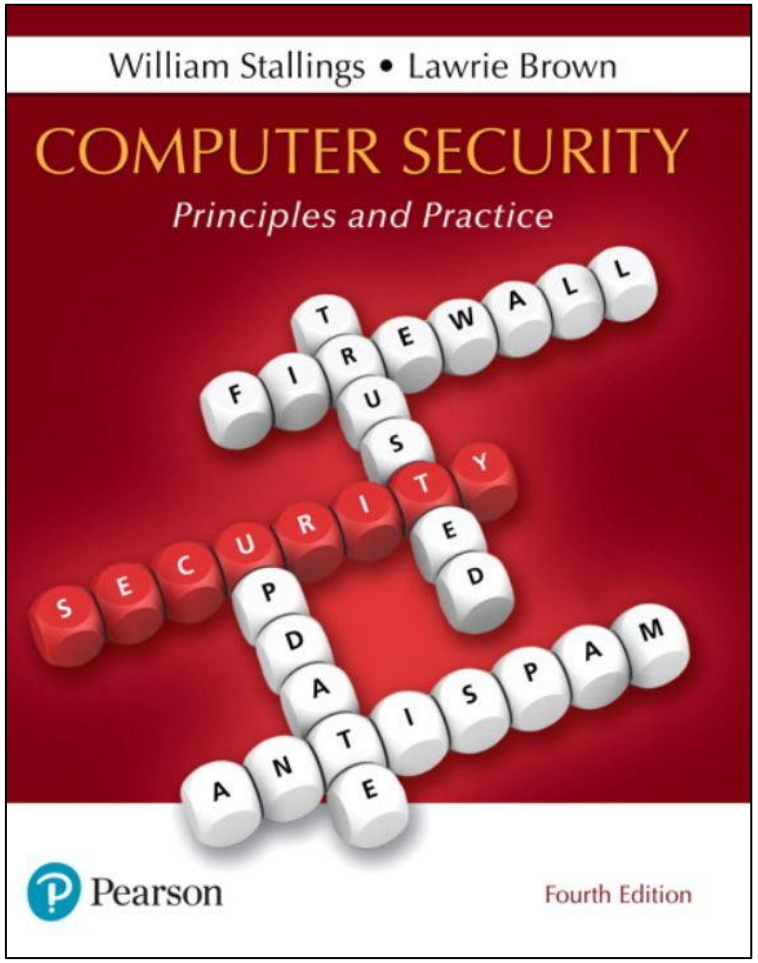
Basics of cryptography – part 1

Dr. Valerio Formicola

CalPolyPomona

Computer Security: Principles and Practice

Fourth Edition



Chapter 2

Cryptographic Tools

Pearson

Cryptography

It's a technic that can be used in various ways to provide all CIA properties, Confidentiality, Integrity and Availability.

Techniques used in cryptography with different purposes:

- *Symmetric encryption*
- *Secure message hashing (hash functions)*
- *Asymmetric encryption*

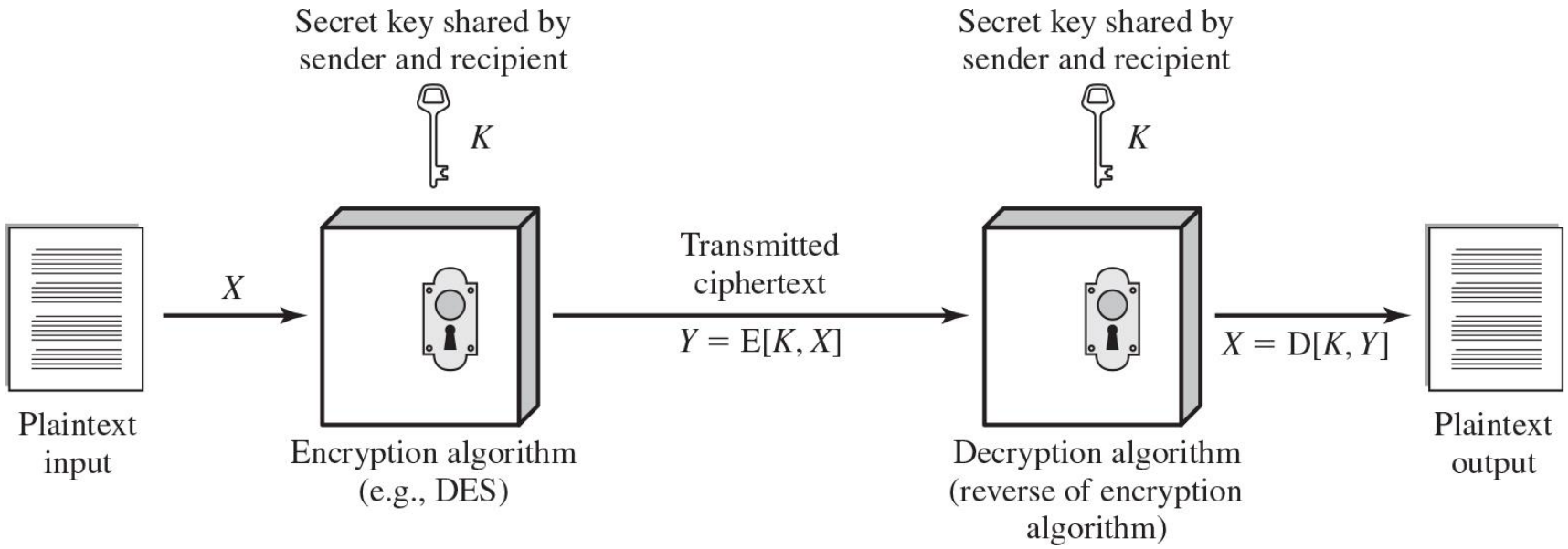
Confidentiality with symmetric encryption

CalPolyPomona

Confidentiality

- It can be achieved if we avoid anybody able to observe a message, is still not able to understand the content of the message
  - e.g., the attacker might captur it on a transmission medium, like cable or air (type of attack called as *eavesdropping* attack, a type of passive attack)

Figure 2.1 Simplified Model of Symmetric Encryption



Note: a plaintext message is not necessarily text. It is anything that you can use with proper decoding; for example, a text with letters or a multimedia forma, or simply commands or instructions

Symmetric Encryption

- The universal technique for providing confidentiality for transmitted or stored data
- Also referred to as conventional encryption or *single-key encryption*
- Two requirements for secure use:
  - Need a strong encryption algorithm
  - Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure

What can be attacked in such a schema?

- The attacker and anybody else usually knows how the encryption/decryption algorithm works, so that's not a secret
  - Algorithms are mostly standard and public

- The attacker might very well intercept and study the encrypted messages
  - that's why we need a mechanism that it doesn't allow to reverse the process

Attacking Symmetric Encryption

Cryptanalytic Attacks

- Rely on:
  - Mathematical “nature” of the algorithm
  - Some knowledge of the general characteristics of the plaintext
    - e.g., all emails start with Hello X …
  - Ideal for the crypto-analyst: Some sample plaintext-ciphertext pairs (aka, *cleartext* analysis)
- Exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or the key being used
  - If successful, all future and past messages encrypted with that key are compromised

Brute-Force Attacks

- Try all possible keys on some ciphertext until an intelligible translation into plaintext is obtained
  - On average half of all possible keys must be tried to achieve success
  - An attacking tool shows results of decryption with a wrong key

A cryptographic algorithm is said to be **breakable** if a crypto-analyst can systematically recover the original message without knowing the key

- Example of attack in cryptoanalysis: **frequency analysis** (also known as **counting letters**) is the study of the frequency of letters or groups of letters in a ciphertext.
- Example of attack in cryptoanalysis: **Dictionary attack** to use names and common words with small substitutions; **credential stuffing** to use databases of users and passwords on different accounts.

Table 2.1 Comparison of Three Popular Symmetric Encryption Algorithms

Block ciphers: The most commonly used symmetric encryption algorithms.
A block cipher processes the plaintext input in fixed-size blocks and produces a block of ciphertext of equal size for each plaintext block

	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

Data Encryption Standard (DES)

- Until recently was the most widely used encryption scheme
  - FIPS PUB 46
  - Referred to as the Data Encryption Algorithm (DEA)
  - Uses 64 bit plaintext block and 56 bit key to produce a 64 bit ciphertext block

- Strength concerns:
  - Concerns about the algorithm itself
    - DES is the most studied encryption algorithm in existence.
      - Quite stable in practice but very old (i.e., very studied)
  - Concerns about the use of a 56-bit key
    - The speed of commercial off-the-shelf processors makes this key length woefully inadequate

Table 2.2 Average Time Required for Exhaustive Key Search

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

- Note: book as some error. The number of decryptions are per seconds (not per micro-seconds)

Triple DES (3DES)

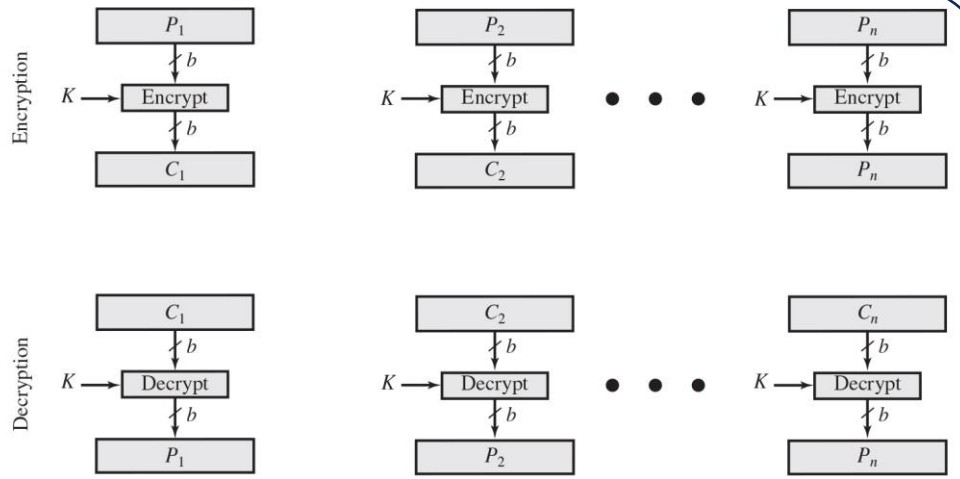
- Repeats basic DES algorithm three times using either two or three unique keys
- First standardized for use in financial applications in ANSI standard X9.17 in 1985
- Attractions:
  - 168-bit key length overcomes the vulnerability to brute-force attack of DES
  - Underlying encryption algorithm is the same as in DES
- Drawbacks:
  - Algorithm is sluggish in software (slow when implemented)
  - Uses a 64-bit block size
    - Too many fragmentations for larger data

Advanced Encryption Standard (AES)

- Needed a replacement for 3DES
  - 3DES was not reasonable for long term use
- NIST called for proposals for a new AES in 1997
  - Should have a security strength equal to or better than 3DES
  - Significantly improved efficiency
  - Symmetric block cipher
  - 128 bit data and 128/192/256 bit keys
- Selected Rijndael in November 2001
  - Published as FIPS 197
- AES is now widely available in commercial products.

Figure 2.2 Types of Symmetric Encryption

Block Cipher
(in ECB mode)



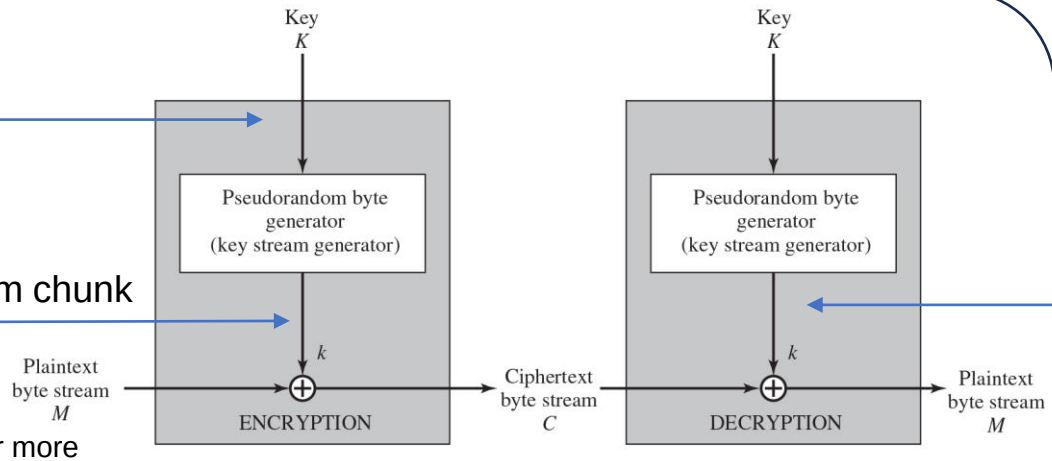
(a) Block cipher encryption (electronic codebook mode)

Stream Cipher

Always the same
key k

Always different
keystream generated for each stream chunk

M can be 1 byte or 1 bit or more



(b) Stream encryption

In Stream cipher
sender and receiver have
the same key.
How do they have
the same keystream?

The principle of stream ciphers is that the sender and the receiver agree on an ***algorithm***,
a **secret key** and ***some parameters***, and both calculate the *keystream* from those parameters.

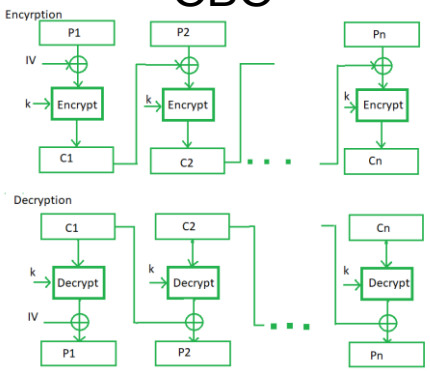
The ***parameters*** can be something like a way to derive a session key from a shared master key or from a key exchange, an IV (a unique value sent at the beginning of each message that allows using the same key for multiple messages).

Practical Security Issues

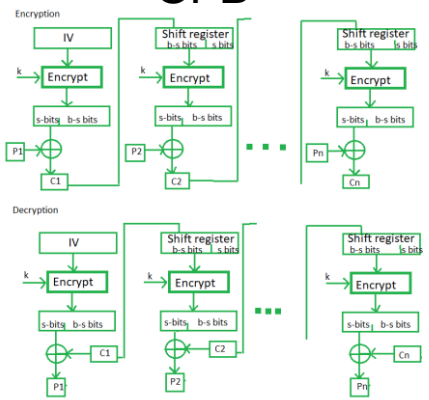
- Typically, symmetric encryption is applied to a unit of data larger than a single 64-bit or 128-bit block
- **Electronic codebook (ECB)** mode is the simplest approach to multiple-block encryption
  - Each block (of same size) of plaintext is encrypted using the same key
  - Cryptanalysts may be able to exploit regularities in the plaintext
    - E.g., the beginning of a message might follow some patterns like a headline in an email, etc.
- Modes of operation
  - Alternative techniques developed to increase the security of symmetric block encryption for large sequences:
    - Cipher Block Chaining (CBC)
    - Cipher Feedback Mode (CFB)
    - Output Feedback Mode (OFB)
    - Counter Mode (CM)
  - Overcomes the weaknesses of ECB

Advantages of using ECB –
•Faster way of encryption in parallel mode.
•Simple way of the block cipher.
Disadvantages of using ECB –
•Prone to cryptanalysis since there is a direct relationship between plaintext and ciphertext.

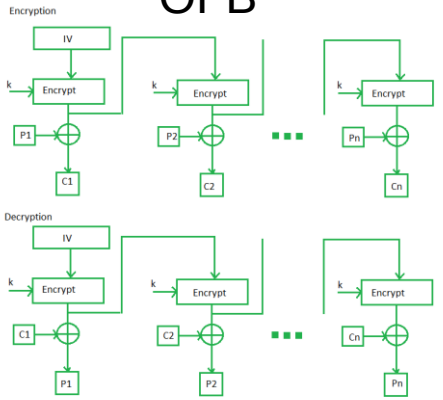
CBC



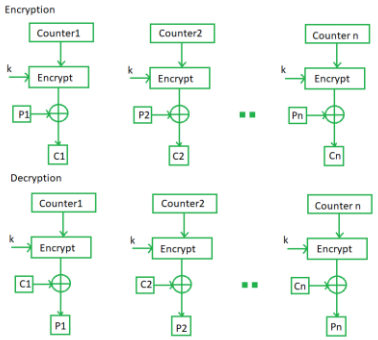
CFB



OFB



CM



Block & Stream Ciphers

- Block Cipher
  - Processes the input one block of elements at a time
  - Produces an output block for each input block
  - Can reuse keys
  - More common

- Stream Cipher
  - Processes the input elements continuously
  - Produces output one element at a time
  - Primary advantage is that they are almost always faster and use far less code
  - Encrypts plaintext one byte at a time
  - Pseudorandom stream is one that is unpredictable without knowledge of the input key and/or initialization parameters

Applications for Block ciphers

1. **Data Encryption:** Block Ciphers are widely used for the encryption of private and sensitive data such as passwords, credit card details and other information that is transmitted or stored for a communication. This encryption process converts a plain data into non-readable and complex form. Encrypted data can be decrypted only by the authorised person with the private keys.
2. **File and Disk Encryption:** Block Ciphers are used for encryption of entire files and disks in order to protect their contents and restrict from unauthorised users. The disk encryption softwares such as BitLocker, TrueCrypt aslo uses block cipher to encrypt data and make it secure.
3. **Virtual Private Networks (VPN):** Virtual Private Networks (VPN) use block cipher for the encryption of data that is being transmitted between the two communicating devices over the internet. This process makes sure that data is not accessed by unauthorised person when it is being transmitted to another user.
4. **Secure Sockets Layer (SSL) and Transport Layer Security (TLS):** SSL and TLS protocols use block ciphers for encryption of data that is transmitted between web browsers and servers over the internet. This encryption process provides security to confidential data such as login credentials, card information etc.
5. **Digital Signatures:** Block ciphers are used in the digital signature algorithms, to provide authenticity and integrity to the digital documents. This encryption process generates the unique signature for each document that is used for verifying the authenticity and detecting if any malicious activity is detected.

Applications for Stream ciphers

- Stream ciphers are often used for their speed and simplicity of implementation in hardware, and in applications where plaintext comes in quantities of unknowable length like a secure wireless connection. If a block cipher (not operating in a stream cipher mode) were to be used in this type of application, the designer would need to choose either transmission efficiency or implementation complexity, since block ciphers cannot directly work on blocks shorter than their block size.
- Mostly used for Real Time applications, e.g., media stream cryptography in DVD/BD players, etc.

Message authentication and hash functions

CalPolyPomona

Message encryption alone cannot provide authentication

- Merely encrypting a message content, does not guarantee that the message is generated by an authentic source.
  - It's only for **confidentiality**
- A malicious actor might intercept the sequence of blocks in ECB encrypted by legitimate user and retransmit the blocks in a different order, hence changing the meaning of the message.
  - (aka, *message reordering attack*)
  - In general, we need more protection against *active attacks*

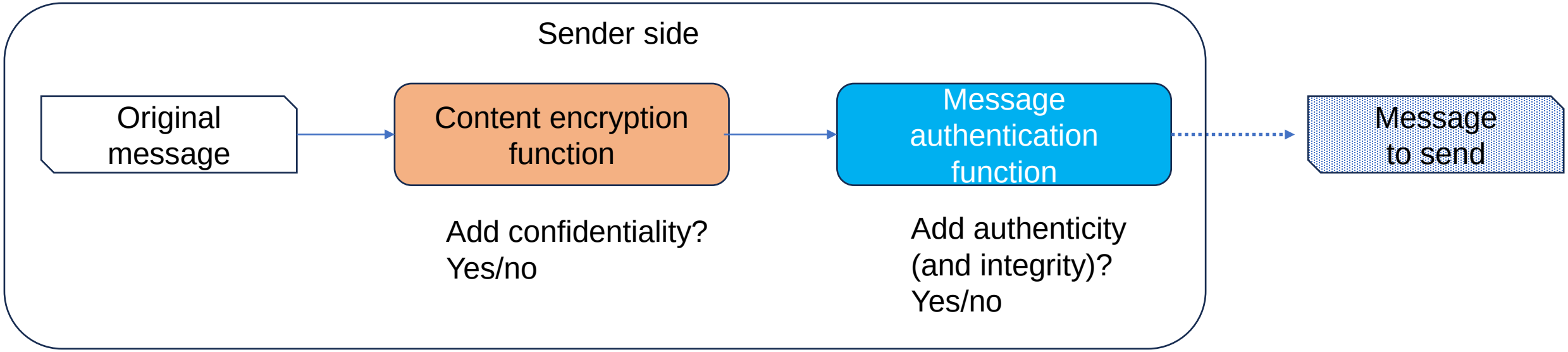
Message Authentication

- We need to add more information to the original message to guarantee it is integer and authentic:
  - Contents have not been altered
  - From authentic source
  - Timely and in correct sequence
- Two mechanisms are usually combined:
  - Integrity: we guarantee that contents of message/data have not been altered
  - Authenticity: data comes from authentic source

- How to obtain: Additional information in a *Message Tag* preceding or following the original message

Observation: Authentication Vs Confidentiality as a functions

- It is possible to combine authentication and confidentiality in a single algorithm by encrypting a message plus its authentication tag
- Typically, message authentication is provided as a separate function from message encryption



Case 1 (not used): Message authentication with symmetric encryption

- Sender and receiver are the only ones having the key and they know each other.
  - Apparently, we might achieve authenticity this way
  - However, …
- Pure encryption of messages does not guarantee:
  - Protection from *reordering attacks*: the attacker simply stays in the middle of a communication and changes the sequence of data messages (1-2-3 -> 3-1-2)
  - Protection from *replay/delay attacks*: the attacker replays packets even if encrypted, to generate the same data at later time

- In practice: symmetric encryption alone is not a suitable tool for data authentication

Case 2 (limited cases): Message Authentication Without Confidentiality

- We guarantee the message is sent from authentic source and it's not altered, but we don't encrypt the content of the message
  - Only authenticity and integrity, but not confidentiality

- Situations in which message authentication without confidentiality may be preferable include:
  - There are a number of applications in which the same message is broadcast to a number of destinations, like alarms or public messages from an authoritative source (e.g., police department)
  - An exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages
    - Only a few messages are encrypted in the content, most likely the ones that contain more critical information
  - Authentication of a computer program in plaintext is an attractive service

Case 3 (used for some cases): authentication with no confidentiality protection, i.e., message is in clear

1. **Approach 1:** Use of plain MAC (Message Authentication Code):
   - Compute a Tag that depends on the content of original message, plus a sequence number (*anti-reordering attack protection*)
   - Encrypt the Tag using a symmetric key (shared key)
   - Append to the original message (in clear)
   - Receiver will use secret key to decrypt the encrypted MAC tag and compare to the actual MAC of the message received
2. **Approach 2:** One-way has function:
   - As before, compute a Tag but this time use a *one-way hash function* of the original message + original length information + padding
   - Encrypt the hash:
     - (2A) Using a symmetric key (shared key)
     - (2B) Using an asymmetric key (aka, public key)
   - Append to the original message (in clear)
   - Receiver will use the key (public or shared) to decrypt the encrypted Hash tag and compare to the actual Hash of message received
3. **Approach 3:** Keyed hash MAC:
   - You evaluate the Hash of a message concatenated to a secret key (shared key): H(K || M || K)
   - Append the keyed Hash tag to the original message
   - Receiver will concatenate the received message with secret key, recalculate the keyed hash and compare with received Hash tag

Approach 1: Message Authentication Using a Message Authentication Code (MAC)

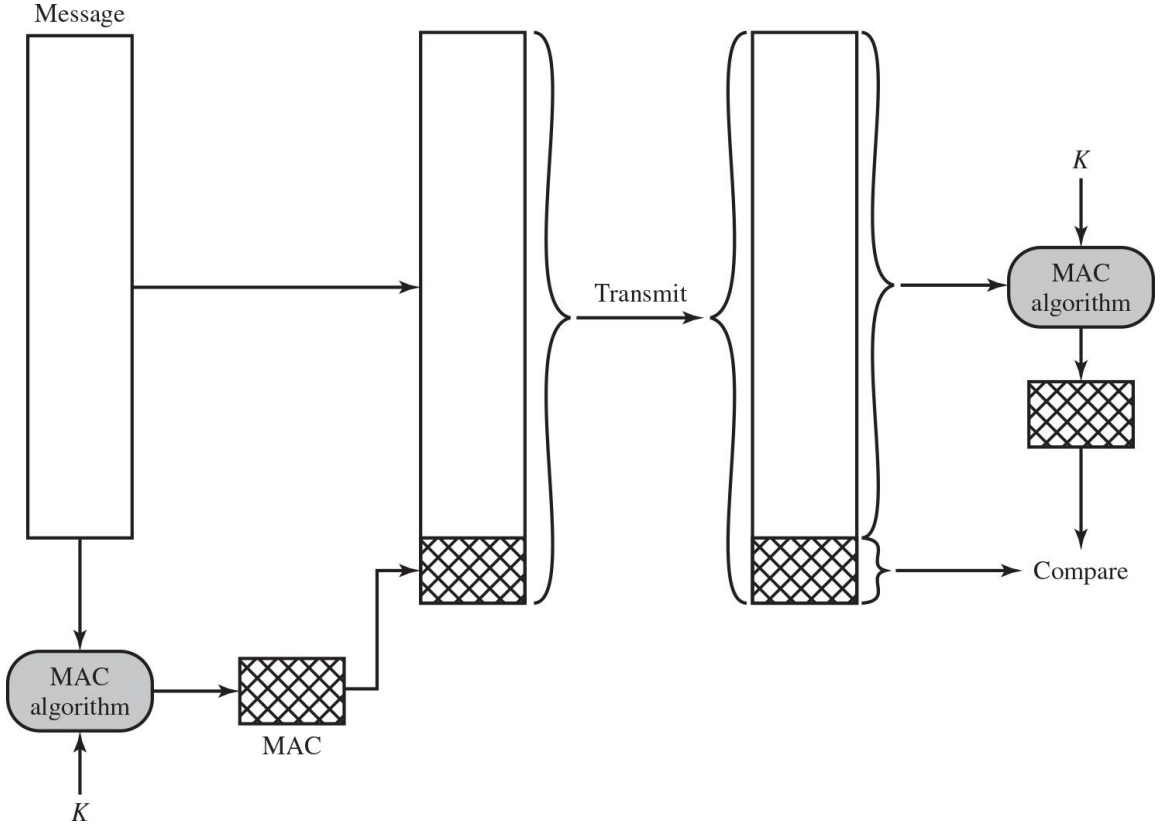
The message authentication code (MAC) is a complex function of the message and the secret key K_{AB} exchanged between sender and receiver: $MAC_M = F(K_{AB}, M)$.

MAC guarantees **integrity** AND **authentication**. Use of a key as an input to the function F guarantees authentication.
Use of a message content M guarantees **integrity**.

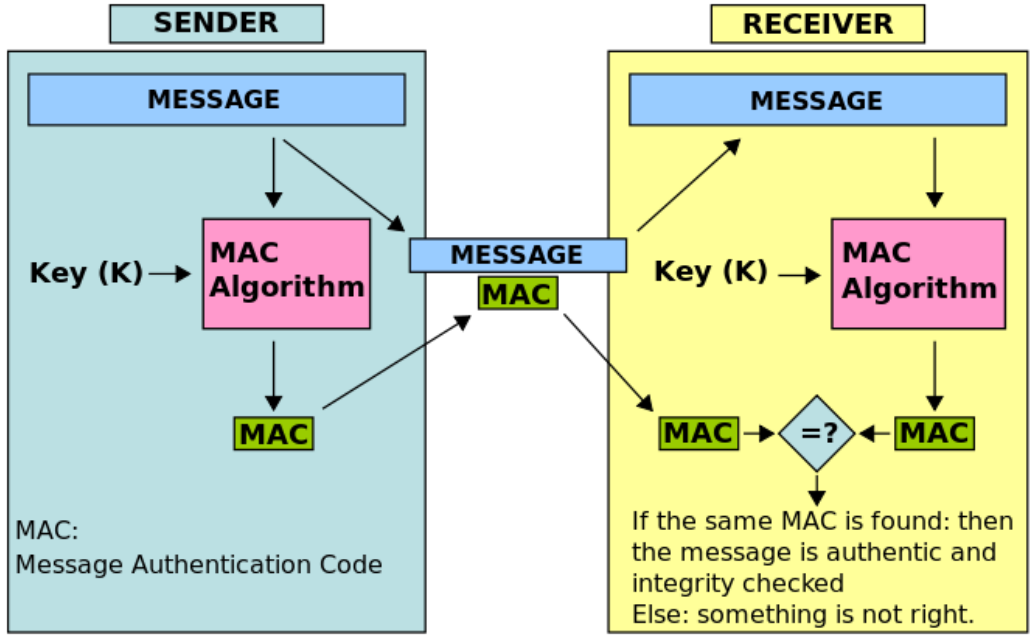
Symmetric key is still used but only for the MAC tag

1 - secret key is used to encrypt a MAC tag (authenticity)
2 - no alteration of message is possible (integrity because nobody has secret key but legitimate ends of the communication)
3 – if the message contains a sequence number, any alteration of the sequence number will be detected as a integrity violation; moreover, sequence number will protect from| *replay attacks* (next slide)

AES is usually used for authentication with MAC.
Recently another called CMAC.



Approach 1 (more)



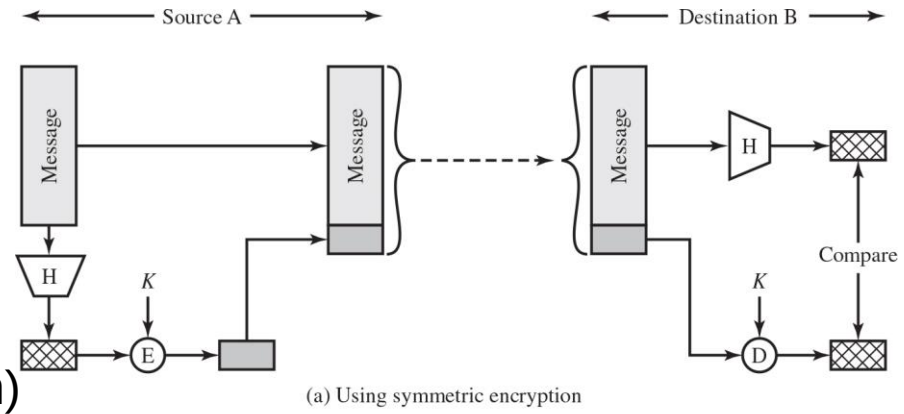
In this example, the sender of a message runs it through a MAC algorithm to produce a MAC data tag. The message and the MAC tag are then sent to the receiver. The receiver in turn runs the message portion of the transmission through the same MAC algorithm using the same key, producing a second MAC data tag. The receiver then compares the first MAC tag received in the transmission to the second generated MAC tag. If they are identical, the receiver can safely assume that the message was not altered or tampered with during transmission (data integrity).

However, to allow the receiver to be able to detect replay attacks, the message itself must contain data that assures that this same message can only be sent once (e.g. time stamp, sequence number or use of a one-time MAC). Otherwise an attacker could – without even understanding its content – record this message and play it back at a later time, producing the same result as the original sender.

Approach 2 (A/B) and Approach 3: Message Authentication Using a One-Way Hash Function

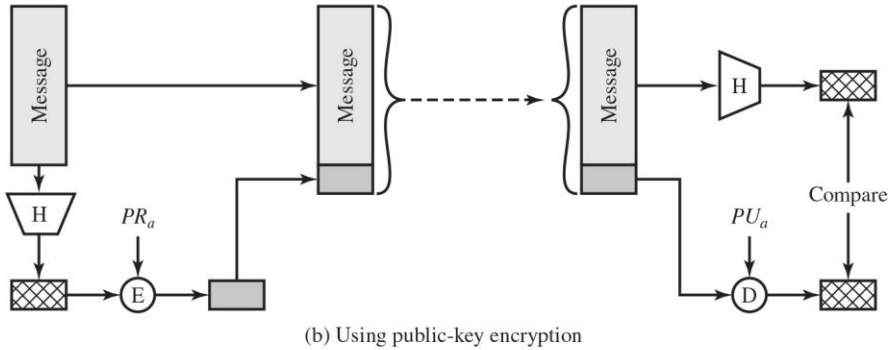
One-way hash functions
aka
Secure Hash functions

Approach 2A
(Symmetric key for encryption)



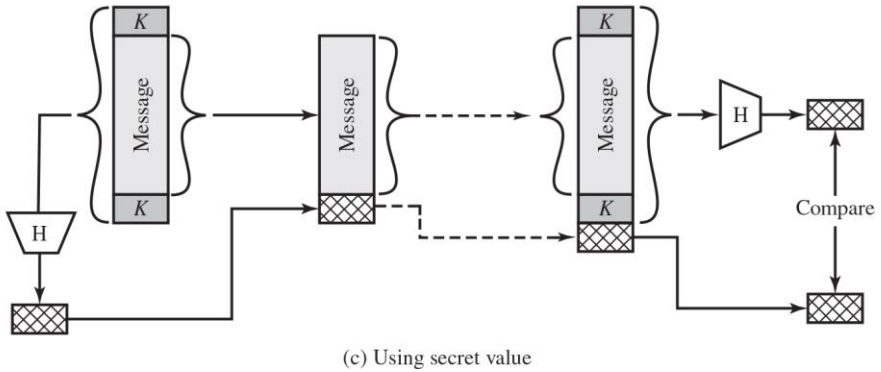
(a) Using symmetric encryption

Approach 2B
(Public key for encryption)



(b) Using public-key encryption

Approach 3
(Keyed-Hash MAC)



(c) Using secret value

Secure Hash Functions: To Be Useful for Message Authentication, a Hash Function H Must Have the Following Properties:

1. Can be applied to a block of data of any size
2. Produces a fixed-length output
3. H(x) is relatively easy to compute for any given *x*
4. **One-way** or **pre-image resistant**
   - Computationally infeasible to find *x* such that H(x) = h, i.e., nobody should be able to find the inverse function of H because, otherwise $H^{-1}(y) = x$;
     x might be K || M || K and the key K would be revealed if you also have M
5. **Second pre-image resistant** or ***weak* collision resistant**
   - Computationally infeasible to find y≠ x such that H(y) = H(x)
   - Note: here you get the hash of a message, but you cannot create another one that has the same hash of the first (*anti-forgery protection*)
6. **Collision resistant or *strong* collision resistance**
   - Computationally infeasible to find any pair (*x*, *y*) such that H(y) = H(x)
   - Note: Here you don't have any starting message; the property states you should not be able to create two messages with the same hash

Security of Hash Functions

- There are two approaches to attacking a secure hash function:
  - Cryptanalysis
    - Exploit logical weaknesses in the algorithm
  - Brute-force attack
    - Strength of hash function depends solely on the length of the hash code produced by the algorithm
- MD5 which generates 128 bit hash, has been found to be breakable and it's not secure anymore
- SHA most widely used hash algorithm family. Currently SHA-2 (256. 384, 512 bit hashes) are the most used and SHA-3 will be in the future
- Additional secure hash function applications:
  - Passwords
    - Hash of a password is stored by an operating system
  - Intrusion detection
    - Store H(File) for each file on a system and secure the hash values

In summary

- MAC function:
  - Calculates an output Tag from the message in combination with a key
  - Hence provides integrity and authenticity
- One-way hash function or Secure-hash function:
  - message Tag is calculated only using the message content, not a key
  - Hence, it provides integrity but not authenticity
  - You need to combine with encryption for authenticity (symmetric or public encryption) to obtain authenticity
  - It has applicable to any blocks size
    - In contrast to stream ciphers that cannot be applied to blocks shorter than the key size
  - Generates the same output, whatever the input size
  - It's not computationally stressful
  - It can have several levels of security based on the resistance level:
    - preimage resistance: for essentially all pre-specified outputs, it is computationally infeasible to find any input that hashes to that output; i.e., given y, it is difficult to find an x such that $h(x) = y$.
    - second-preimage resistance: for a specified input, it is computationally infeasible to find another input which produces the same output; i.e., given x, it is difficult to find a second input $x' \neq x$ such that $h(x) = h(x')$.
    - collision resistance (strong resistance): it is computationally infeasible to find any two distinct inputs x, x′ that hash to the same output; i.e., such that $h(x) = h(x')$