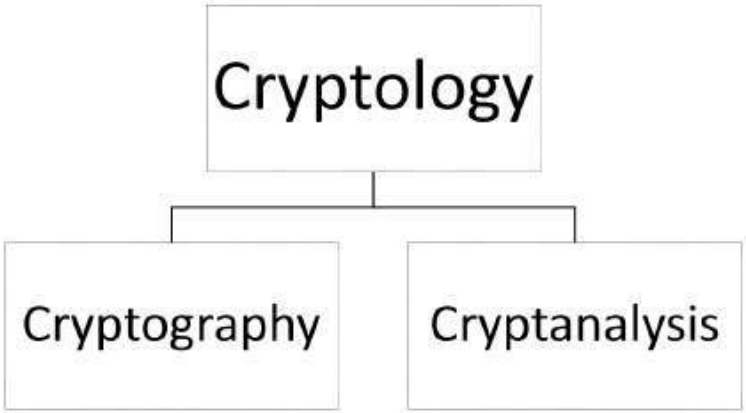
# Module 2
# Security Threats & Countermeasures

## Submodule 2: Introduction to Cryptography

# Cryptology

- Cryptology is the study of cryptosystems.
- It can be divided into two branches:
  - Cryptography
    - Make a cryptosystem that can provide information security
  - Cryptanalysis
    - Art and science to break the cipher text



# What is Cryptography?

- Cryptography is the practice and study of techniques for secure communication in the presence of third parties called adversaries.

- Cryptography helps to achieve information security by enforcing:
  - Confidentiality
  - Data Integrity
  - Authentication
  - Non-repudiation

Exactly, how cryptography help to achieve each of them?

# History of Cryptography

- Can be traced back to 4,000 years ago when ancient Egyptians use “hieroglyph” to transmit secret messages.
- Romans used Caesar Shift Cipher.
- Steganography.
- Vigenere coding.
- Enigma rotor machine.
- …

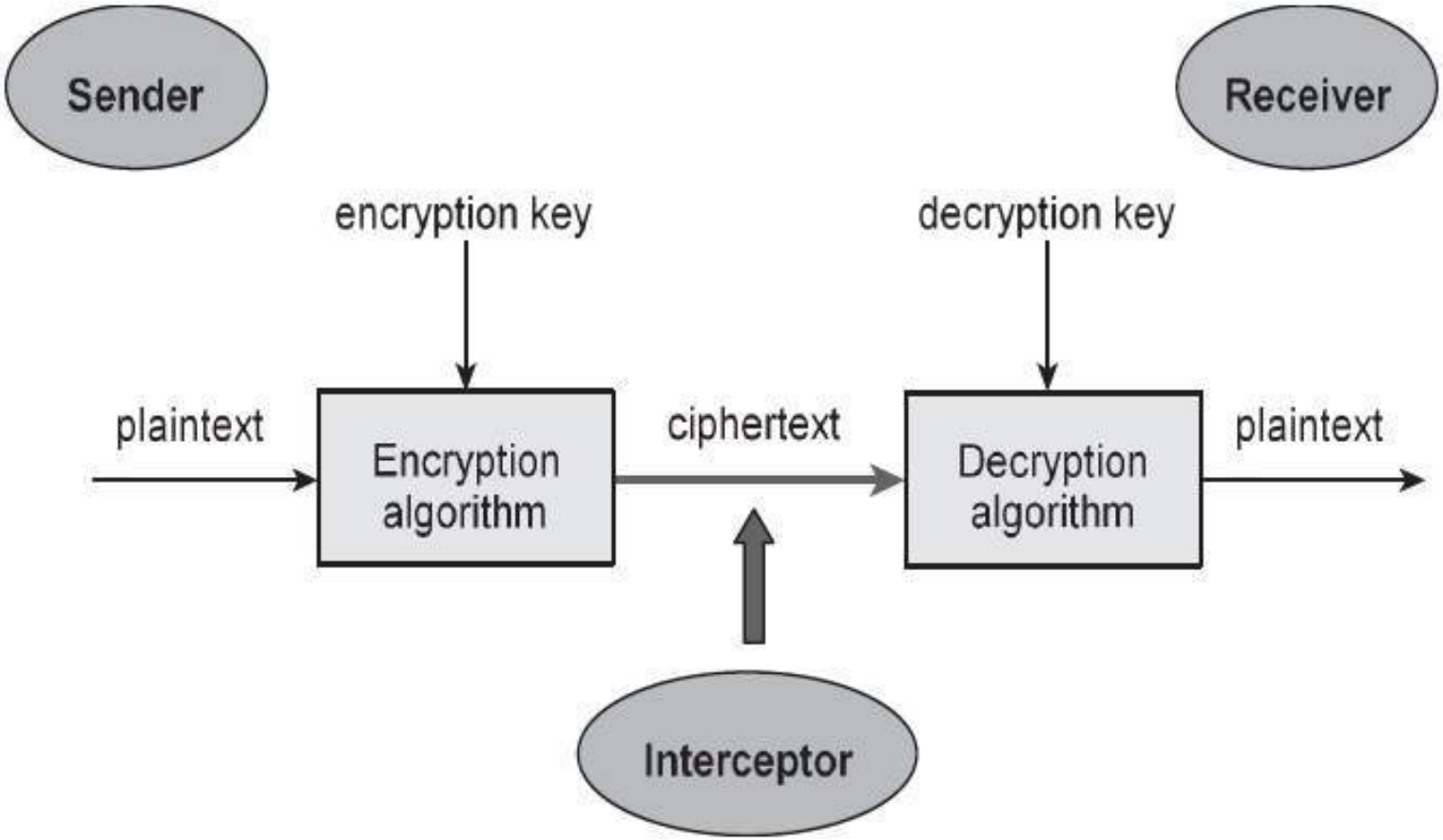
# Cryptography Primitives

- The tools and techniques that can be selectively used to provide a set of desired security services:
  - Encryption
  - Hash functions
  - Message Authentication Codes (MAC)
  - Digital signature
- These cryptographic primitives are related and often combined to achieve a set of desired security goals.

| | Encryption | Hash Function | MAC | Digital Signature |
|---|---|---|---|---|
| **Confidentiality** | Yes | No | No | No |
| **Integrity** | No | Sometimes | Yes | Yes |
| **Authentication** | No | No | Yes | Yes |
| **Non Repudiation** | No | No | Sometimes | Yes |

# Cryptographic Concepts

- Encryption: establish confidential communication in an unsecure environment.
  - C = E (M)—The process of encryption using encryption algorithm.
  - M = D(C)—The process of decryption using decryption algorithm.
  - What relationship should exist between the encryption and decryption algorithms?


Sender
Receiver
encryption key
decryption key
plaintext
Encryption algorithm
ciphertext
Decryption algorithm
plaintext
Interceptor

# Cryptosystems

- Seven components of a cryptosystem:
  - The set of possible plaintexts
  - The set of possible ciphertexts
  - The set of encryption keys
  - The set of decryption keys
  - The correspondence between encryption keys and decryption keys
  - The encryption algorithm to use
  - The decryption algorithm to use
- Two types of cryptosystems:
  - Symmetric key encryption
  - Asymmetric key encryption

$$c_i = E(p_i) = p_i + 3$$

A full translation chart of the Caesar cipher is shown here.

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Plaintext** | 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 |
| **Ciphertext** | 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 |

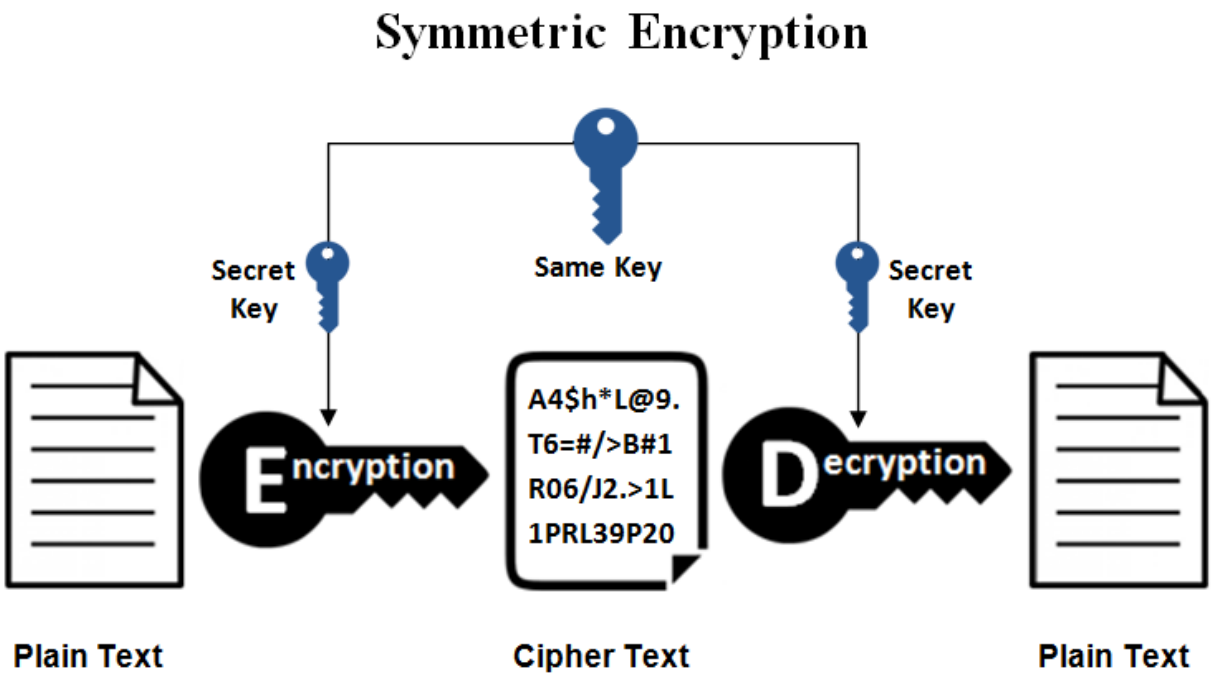
Using this encryption, the message

TREATY IMPOSSIBLE

would be encoded as

| T | R | E | A | T | Y | | I | M | P | O | S | S | I | B | L | E |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| w | u | h | d | w | b | | l | p | s | r | v | v | l | e | o | h |

# Symmetric Cryptosystems



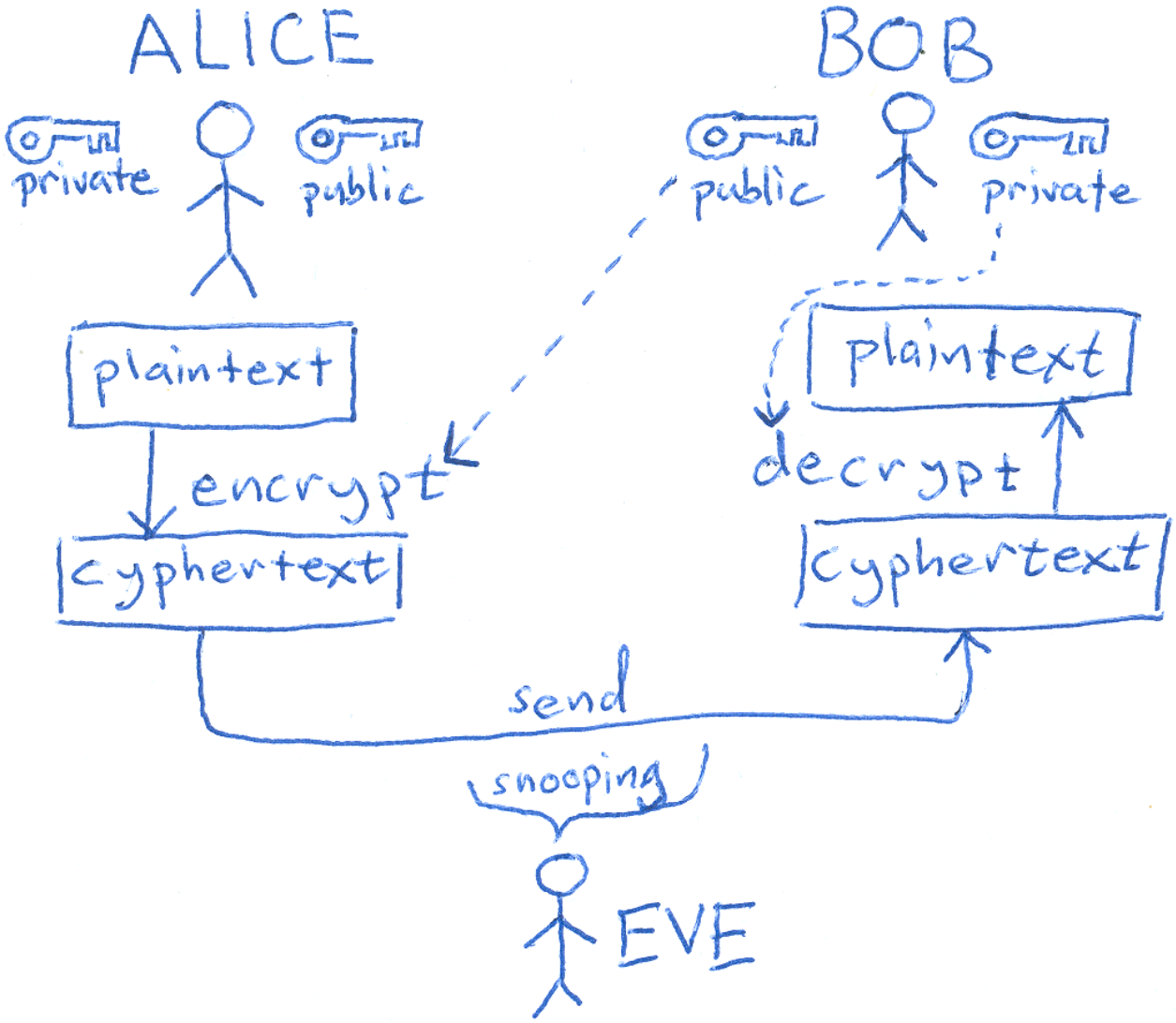
# Features of Symmetric Encryption

- How to share the secret key, securely?
  - The key must be shared prior to communication.
  - Keys are often recommended to be changed regularly.
- Large number of secret keys to manage:
  - For instance, how many secret keys are needed to enable a party of n users to use symmetric cryptosystem for the purpose of confidential communication—the answer is n x (n-1) / 2.
- Length of key is smaller—process of encryption-decryption is faster. Required computing power is less.

# Asymmetric Key Encryption

- Different keys are used for encrypting and decrypting.
  - Private key and public key—that are mathematically related. However, it is computationally not feasible to figure out one from another.
  - Public key goes to a public repository and private key remains secret.
  - Keys are of large length—the encryption-decryption process is slower and resource-demanding.

# Public Key Cryptosystem



# Public Key Cryptosystems

- Advantages:
  - No need to worry about sending the shared symmetric key to both ends of the communication.
  - Only need to keep private key secret.
  - The number of keys required is substantially reduced:
    - How many keys to maintain for n users?
- Any problem?
  - Encryption and decryption algorithms tend to perform slower than symmetric encryption.
  - Key length is bigger compared to symmetric key.

# Think It Through

- Bob—has two keys $P_B$ and $S_B$
  - $C = E_{PB}(M)$—Encrypting the message
    - Who can do this?
  - $M = D_{SB}(C)$—decrypting the message
    - Who can do this?
- $S = D_{SB}(M)$ and $M = E_{PB}(S)$ or $E_{PB}(D_{SB}(M))=M$
  - We can reverse the order of encryption and decryption
  - How can this be useful?
  - What does this provide?
  - The idea behind digital signature.

# Kerckhoff's Design Principles of Cryptosystem

- The cryptosystem should be unbreakable practically, if not mathematically.

- Falling of the cryptosystem in the hands of an intruder should not lead to any compromise of the system, preventing any inconvenience to the user —the Kerckhoff principle

- The key should be easily communicable, memorable, and changeable.

# Kerckhoff's Design Principles of Cryptosystem

- The ciphertext should be transmissible by telegraph, and unsecure channel.

- The encryption apparatus and documents should be portable and operable by a single person.

- It is necessary that the system be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observer.

# Attacks on Cryptosystems

- Based on the action performed by the attacker toward the cryptosystem, an attack could be:
  - Passive attack
    - The goal is to gain unauthorized access to the information, such as intercepting and eavesdropping
    - This type of attack does not affect the information
    - This type of attack does not disrupt the communication channel
  - Active attack
    - Involves changing the information in some way by conducting some process on the information
      - Modify the information
      - Alter of authentication data
      - Unauthorized deletion of data

# Cryptosystem Attack Methods-I

- Ciphertext only attacks
  - Attacker has access to a set of ciphertexts(s) but not corresponding plaintext.
  - Succeed if corresponding plaintext can be determined from a given set of ciphertext.
  - Occasionally, key can be determined.
- Known plaintext attacks
  - Attacker knows the plaintext for some parts of the ciphertext.
  - Goal is to decrypt the rest of the ciphertext using the known information.
  - Can be done by determining the key or some other ways.

# Cryptosystem Attack Methods-II

- Chosen plaintext attacks
  - Attacker has the ciphertext-plaintext pair of his choice.
  - Makes determining the encryption key easier.
- Dictionary attack
  - Compiling a “dictionary” of ciphertexts and corresponding plaintexts
  - Refer to the “dictionary” to find corresponding plaintext of obtained ciphertext.
- Brute Force attack
  - Try to determine the key by attempting all possible keys.
  - The attacker has access to the ciphertext and the algorithm.

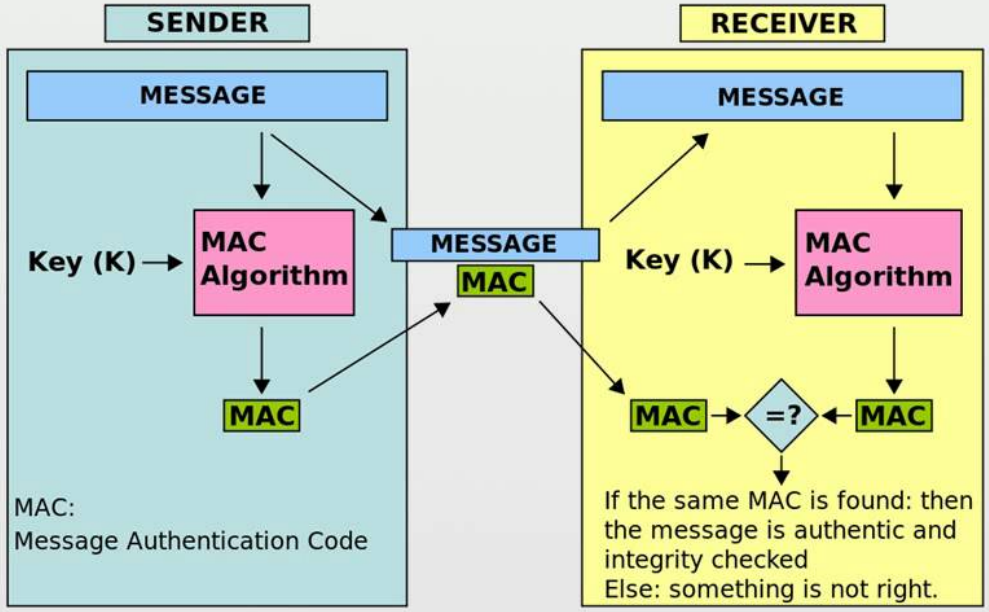
# Cryptosystem Attack Methods-III

- Birthday attack
  - Used against hash function.
  - The goal is to break the hash function by finding a collision.
- Man-in-the-Middle attack
  - Often target public key cryptosystems
    - Host A wants to communicate to host B, hence requests public key of B
    - An attacker intercepts this request and sends his public key instead—whatever A sends to B, the attacker can read.
    - Attacker re-encrypts the data after reading with his public key and sends to B.
    - The attacker sends his public key as A's public key so that B takes it as if it is taking from A.

# Hash Function

- Cryptographic hash functions
  - allow one to easily verify that some input data maps to a given hash value
  - Should be one-way function: easy to compute but hard to invert.
    - Use case—making digital signature easier and man-in-the-middle attack harder:
      - Bob computes signature as $S = E_{SB}(h(M))$
      - Alice computes h(M) and checks to see if $D_{PB}(S) = h(M)$
      - h(M) is called digest of M
  - Should be collision resistant—not likely to have h(M) = h(M')
  - May also provide integrity—how?

# Message authentication code




https://en.wikipedia.org/wiki/File:MAC.svg

# Digital Certificates

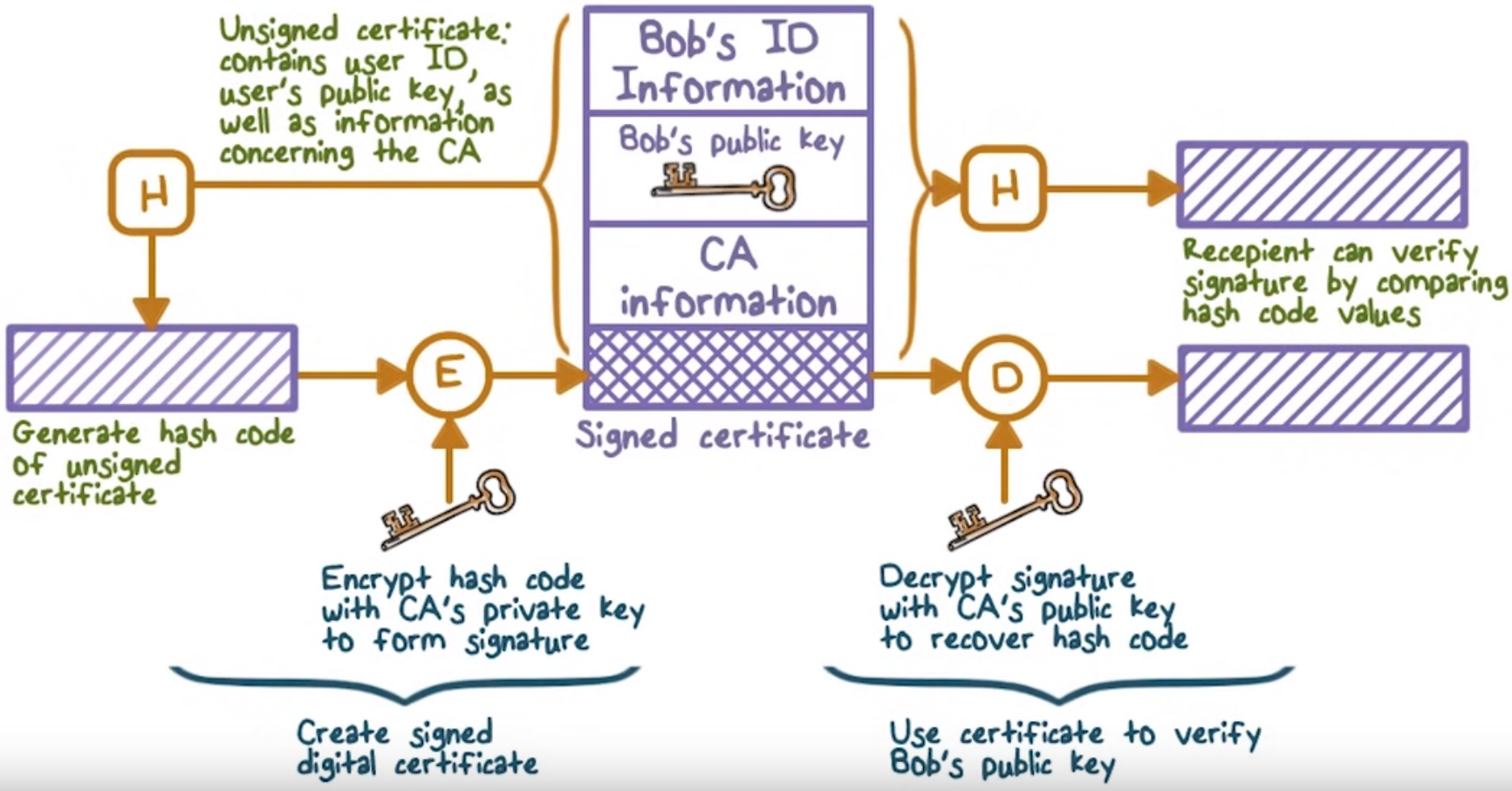
- Entrust an authority (Certificate Authority CA) that can digitally sign a statement to verify that a person indeed owns his/her public key—i.e., they are who they claim they are.

- Digital certificate combines public key with identifying information.

- A digital certificate should contain the following information:
  - Name of the certification authority
  - Date of issuance of the certificate
  - Expiration date of certificate
  - Address of the website
  - Name of the organization operating the web site
  - Public key used of the web server
  - Name of the cryptographic has functions used
  - Digital signature

Check out a digital certificate in IE: Tools→Internet Option→Content →Certificates

# Public Key Certificate



# Obtaining/Using a Digital Certificate

- Subscriber generates a public/private key pair. Applies to CA for digital certificate with the public key.
- CA verified subscriber's identity and issues digital certificate containing the public key.
- CA publishes certificate to public, online repository.
- Subscriber signs message with private key and sends message to second party.
- Receiving party verifies digital signature with sender's public key and requests verification of sender's digital certificate from CA's public repository
- Repository reports status of subscriber's certificate.

# Reasons to Use Digital Certificate

- Proving the identity of the sender of a transaction.
- Non-repudiation—the owner of the certificate cannot deny partaking in the transaction.
- Encryption and checking the integrity of data—provide the receiver with the means to encode a reply.
- Single Sign-on—It can be used to validate a user and log them into various computer systems without having to use a different password for each system.

# Cryptanalysis

- Cryptanalysis is the act of attacking cryptosystems in order to discover the plaintext or key.
  - The assumption is that the encryption and decryption algorithms are known to the cryptanalysts, which follows the open design principle.

- Different types of cryptanalytic attacks:
  - Ciphertext-only attack
  - Known-plaintext attack
  - Chosen-plaintext attack
  - Chosen-ciphertext attack
  - Chosen text attack

| Attack Type | Knowledge Known to Cryptanalyst |
|---|---|
| Ciphertext only | • Encryption algorithm<br>• Ciphertext to be decoded |
| Known Plaintext | • Encryption algorithm<br>• Ciphertext to be decoded<br>• One or more plaintext-ciphertext pairs formed with the same secret key |
| Chosen Plaintext | • Encryption algorithm<br>• Ciphertext to be decoded<br>• Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the same secret key |
| Chosen Ciphertext | • Encryption algorithm<br>• Ciphertext to be decoded<br>• Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key |
| Chosen text | • Encryption algorithm<br>• Ciphertext to be decoded<br>• Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key<br>• Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key |

# Substitution Ciphers

- Each letter is uniquely replaced by another.
  - There are 26! possible substitution ciphers—Why?
  - There are more than $4.03 \times 10^{26}$ such ciphers.

- Letters in a natural language, like English, are not uniformly distributed.
  - Knowledge of letter frequencies, including pairs and triples can be used in cryptologic attacks against substitution ciphers—How?

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| a: | 8.05% | b: | 1.67% | c: | 2.23% | d: | 5.10% |
| e: | 12.22% | f: | 2.14% | g: | 2.30% | h: | 6.62% |
| i: | 6.28% | j: | 0.19% | k: | 0.95% | l: | 4.08% |
| m: | 2.33% | n: | 6.95% | o: | 7.63% | p: | 1.66% |
| q: | 0.06% | r: | 5.29% | s: | 6.02% | t: | 9.67% |
| u: | 2.92% | v: | 0.82% | w: | 2.60% | x: | 0.11% |
| y: | 2.04% | z: | 0.06% | | | | |

**8.1:** Letter frequencies in the book *The Adventures of Tom Sawyer*, by Twain.

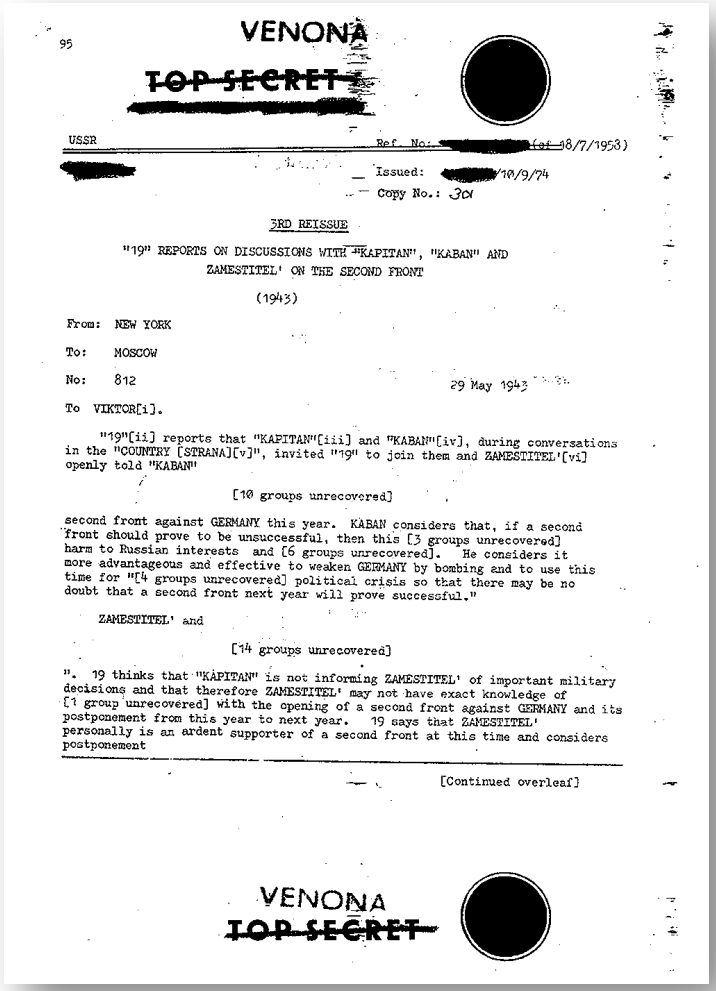
# One-Time Pads

- One type of substitution cipher could be unbreakable.
  - The **one-time pad** was invented in 1917 by Joseph Mauborgne and Gilbert Vernam.
  - It uses a block of shift keys, ($k_1$, $k_2$, . . . , $k_n$), to encrypt a plaintext, M, of length n, with each shift key being chosen uniformly at random.
    - The length of block of keys has to be same as the length of the plaintext.
    - Each shift amount, i.e., k value must be chosen completely at random.

- Since each shift is random, every ciphertext is equally likely for any plaintext.

# Weakness of the One-Time Pads

- In spite of their perfect security, one-time pads have some weaknesses.
- The key has to be as long as the plaintext.
- Keys can never be reused
  - Repeated use of one-time pads allowed the U.S. to break some of the communications of Soviet spies during the Cold War.



95

VENONA

~~TOP SECRET~~ [redacted]

USSR Ref. No.: [redacted] (of 8/7/1953)

[redacted] Issued: [redacted]/10/9/74

Copy No.: 301

3RD REISSUE

"19" REPORTS ON DISCUSSIONS WITH "KAPITAN", "KABAN" AND ZAMESTITEL' ON THE SECOND FRONT

(1943)

From: NEW YORK

To: MOSCOW

No: 812 29 May 1943

To VIKTOR[i].

"19"[ii] reports that "KAPITAN"[iii] and "KABAN"[iv], during conversations in the "COUNTRY [STRANA][v]", invited "19" to join them and ZAMESTITEL'[vi] openly told "KABAN"

[10 groups unrecovered]

second front against GERMANY this year. KABAN considers that, if a second front should prove to be unsuccessful, then this [3 groups unrecovered] harm to Russian interests and [6 groups unrecovered]. He considers it more advantageous and effective to weaken GERMANY by bombing and to use this time for "[4 groups unrecovered] political crisis so that there may be no doubt that a second front next year will prove successful."

ZAMESTITEL' and

[14 groups unrecovered]

". 19 thinks that "KAPITAN" is not informing ZAMESTITEL' of important military decisions and that therefore ZAMESTITEL' may not have exact knowledge of [1 group unrecovered] with the opening of a second front against GERMANY and its postponement from this year to next year. 19 says that ZAMESTITEL' personally is an ardent supporter of a second front at this time and considers postponement

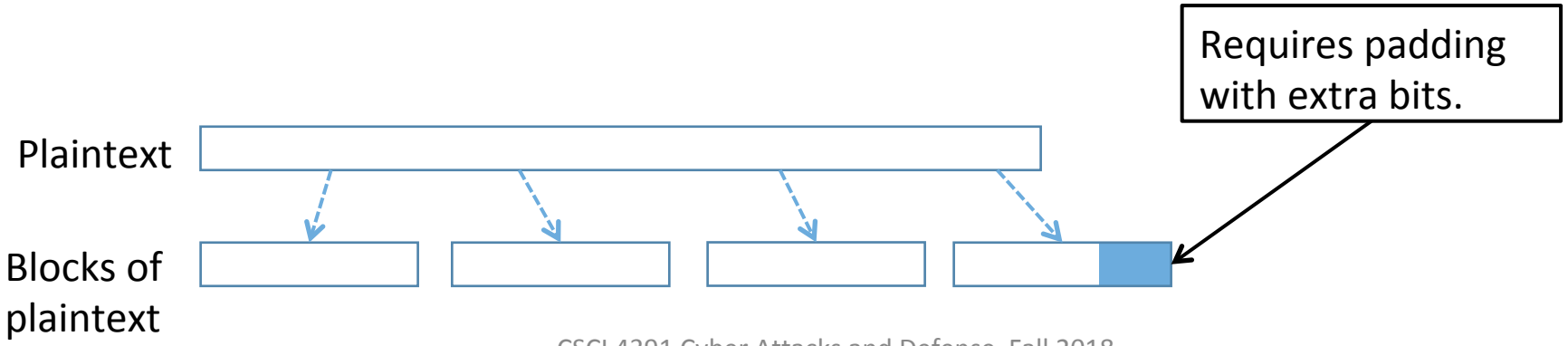
[Continued overleaf]

VENONA

~~TOP SECRET~~ [redacted]

# Block Ciphers

- In a **block cipher:**
  - Plaintext and ciphertext blocks have fixed length b (e.g., 128 bits)
  - A plaintext of length n is partitioned into a sequence of m **blocks**, P[0], …, P[m-1], where $n \leq bm < n + b$
- Each message is divided into a sequence of blocks and encrypted or decrypted in terms of its blocks.



# Block Cipher Basics

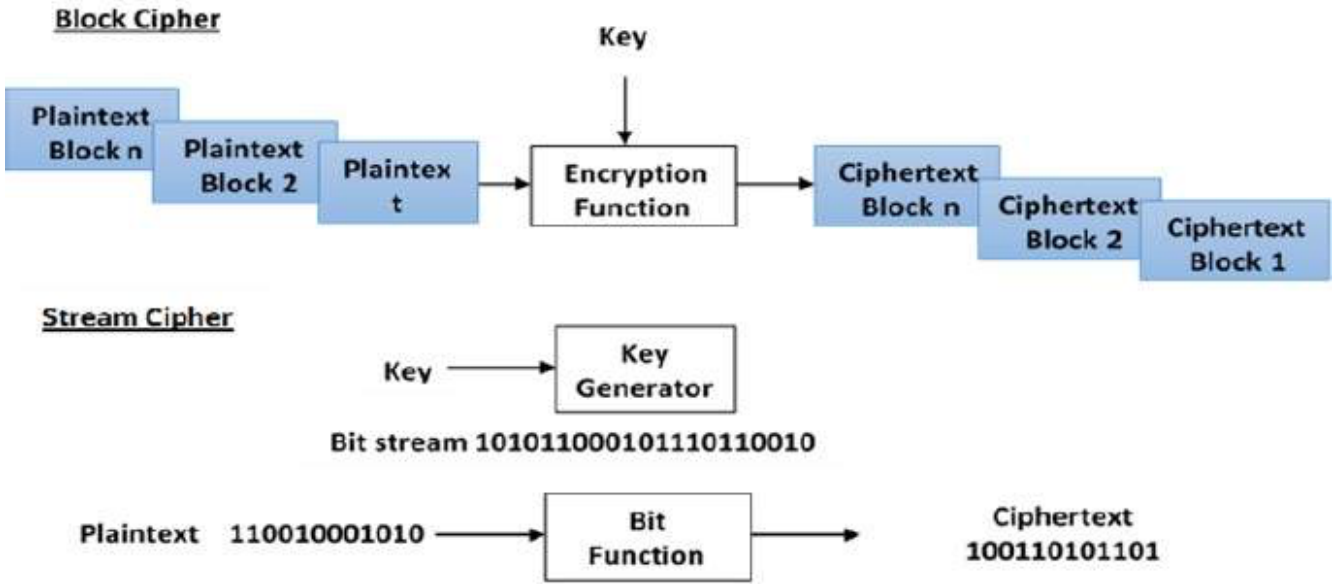
- Block size
  - Avoid very small block size—makes it easier to do dictionary attack
  - Not too big block size—operation efficiency is low
  - Needs to be multiple of 8 bits
- Well-known block cypher schemes
  - DES
  - Triple DES
  - AES
  - IDEA
  - Twofish
  - Serpent

# Padding

- Block ciphers require the length n of the plaintext to be a multiple of the block size b.
- Padding the last block needs to be unambiguous (cannot just add zeroes).
- Example for b = 128 (16 bytes)
  - Plaintext: “Roberto” (7 bytes)
  - Padded plaintext: “Roberto999999999” (16 bytes), where 9 denotes the number and not the character
- We need to always pad the last block, which may consist only of padding.

# Stream Cipher

- The plaintext is processed one bit at a time and a series of operations is performed on it to generate one bit of ciphertext.
- You can think of stream cipher as block cipher with block size of 1 bit.

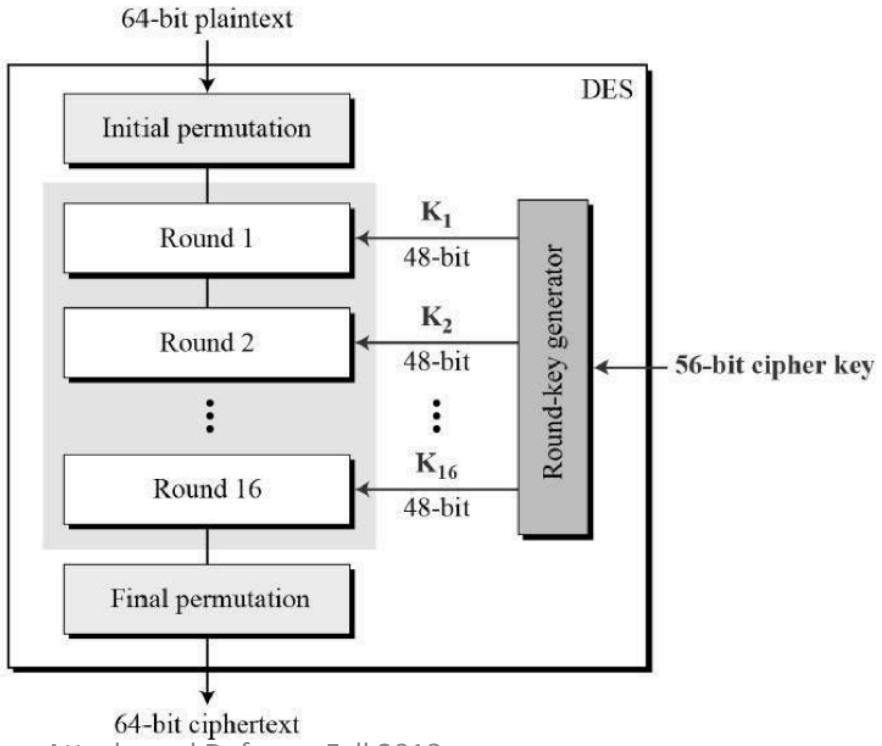


# Stream Cipher

- Key stream
  - Pseudo-random sequence of bits S = S[0], S[1], S[2], …
  - Can be generated on-line one bit (or byte) at the time
- Stream cipher
  - XOR the plaintext with the key stream C[i] = S[i] ⊕ P[i]
  - Suitable for plaintext of arbitrary length generated on the fly, e.g., media stream
- Synchronous stream cipher
  - Key stream obtained only from the secret key K
  - Works for unreliable channels if plaintext has packets with sequence numbers
- Self-synchronizing stream cipher
  - Key stream obtained from the secret key and q previous ciphertexts
  - Lost packets cause a delay of q steps before decryption resumes

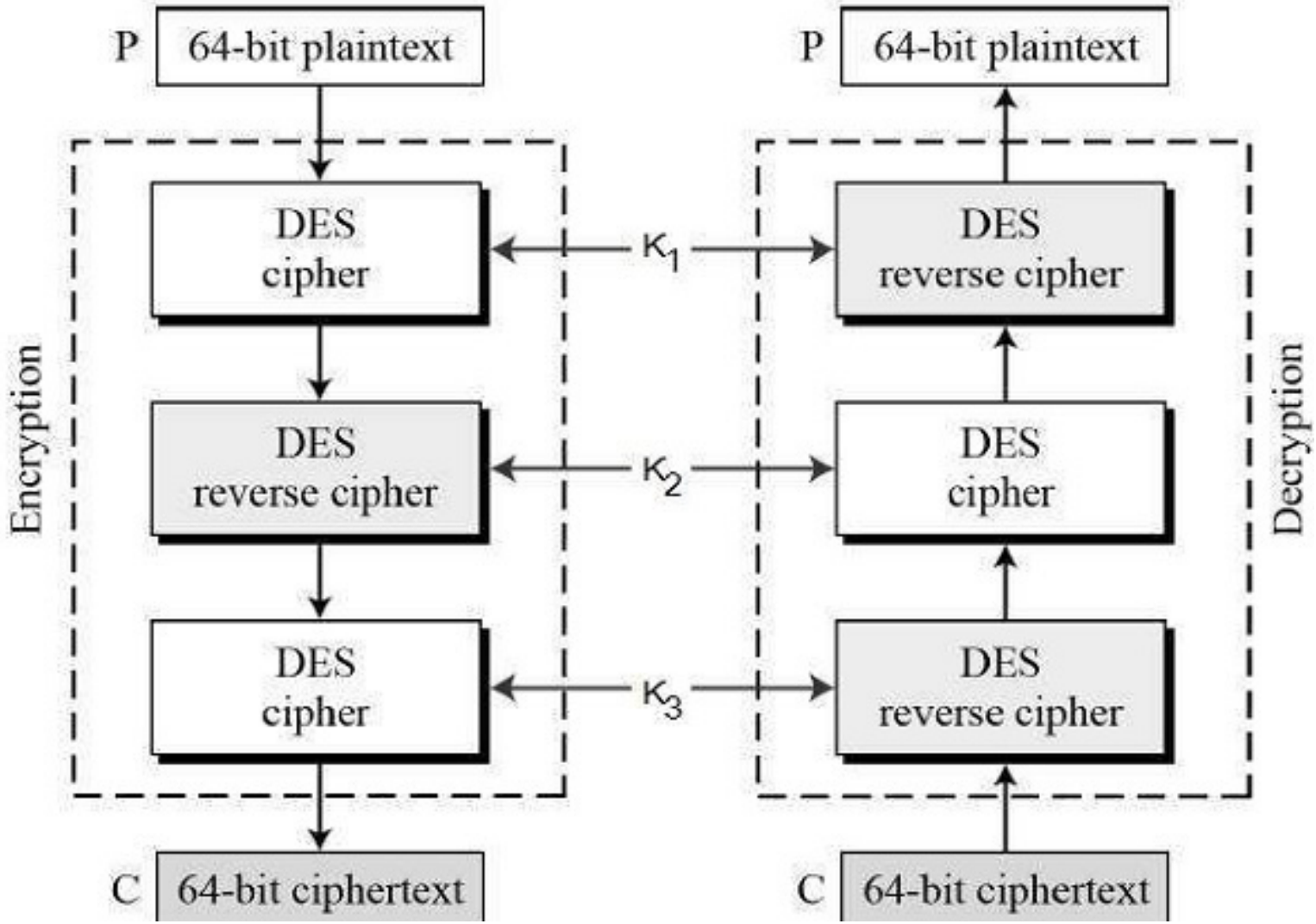
# Block Ciphers-DES

- Data Encryption Standard (DES)
  - Developed by IBM and adopted by NIST in 1977
  - 64-bit blocks and 56-bit keys
  - Small key space makes exhaustive search attack feasible since late 90s



# Block Ciphers-Triple DES

- Triple DES (3DES)
  - Nested application of DES with three different keys KA, KB, and KC
  - Effective key length is 168 bits, making exhaustive search attacks unfeasible
  - $C = E_{KC}(D_{KB}(E_{KA}(P)))$;  $P = D_{KA}(E_{KB}(D_{KC}(C)))$
    - The encrypt-decrypt-encrypt process makes it possible to DES implementation (hardware)
    - Equivalent to DES when KA=KB=KC (backward compatible)

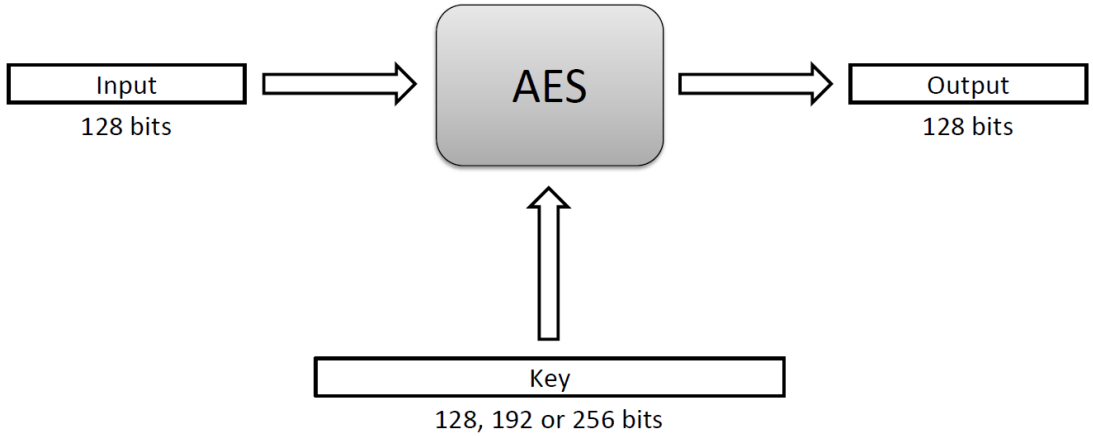

P
64-bit plaintext
DES
cipher
DES
reverse cipher
DES
cipher
C
64-bit ciphertext
K1
K2
K3
Encryption
P
64-bit plaintext
DES
reverse cipher
DES
cipher
DES
reverse cipher
C
64-bit ciphertext
Decryption

# Block Ciphers-II

- Advanced Encryption Standard (AES)
  - Selected by NIST in 2001 through open international competition and public discussion.
  - 128-bit blocks and several possible key lengths: 128, 192 and 256 bits.
  - Exhaustive search attack not currently possible
  - AES-256 is the symmetric encryption algorithm of choice.
  - Software implementation in C and Java.

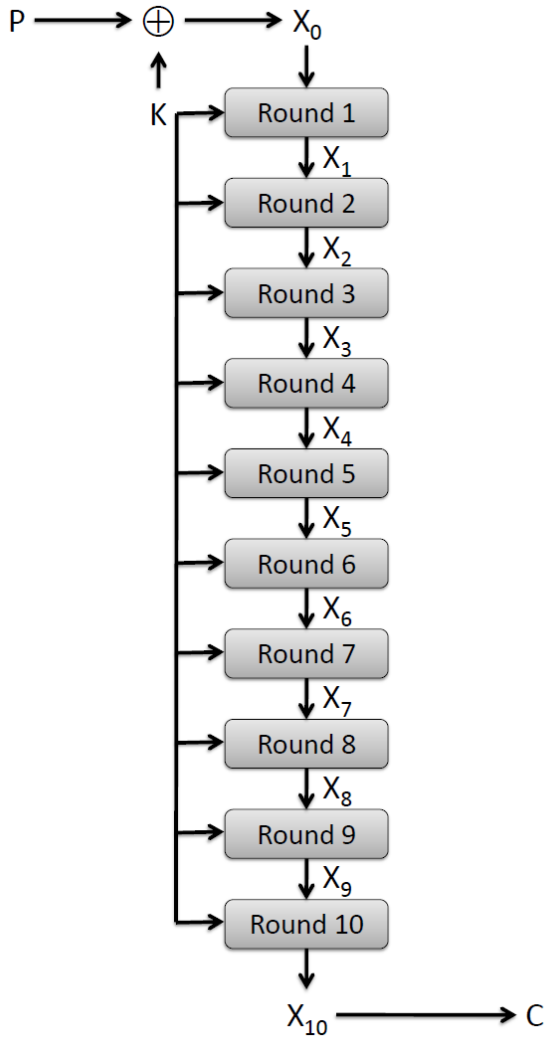
# The Advanced Encryption Standard

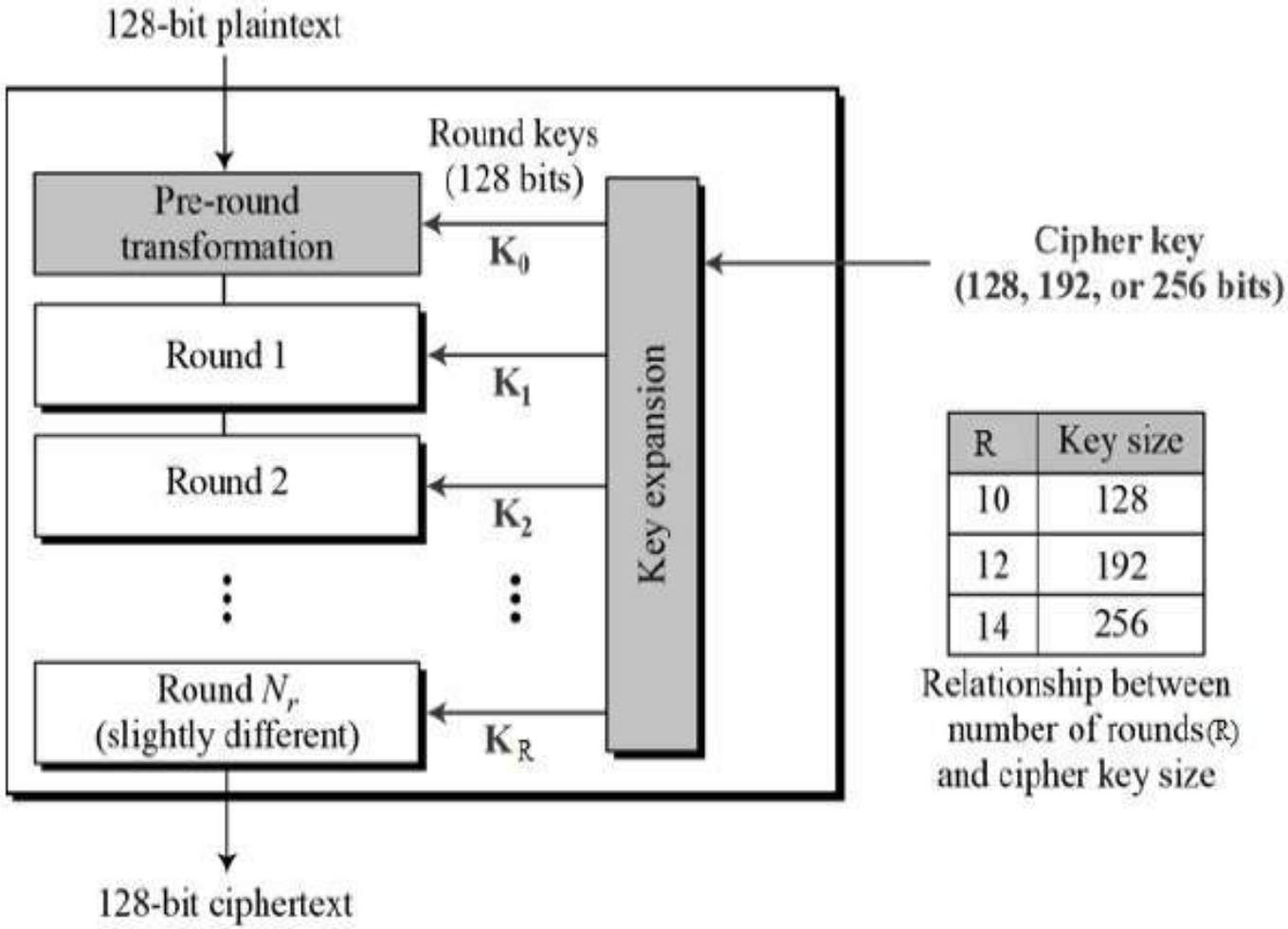
- AES is a block cipher that operates on 128-bit blocks. It is designed to be used with keys that are 128, 192, or 256 bits long, yielding ciphers known as AES-128, AES-192, and AES-256.



# AES Round Structure

- The 128-bit version of the AES encryption algorithm proceeds in ten rounds.
- Each round performs an invertible transformation on a 128-bit array, called **state**.
- The initial state $X_0$ is the XOR of the plaintext P with the key K:
- $X_0 = P \text{ XOR } K.$
- Round i (i = 1, …, 10) receives state $X_{i-1}$ as input and produces state $X_i$.
- The ciphertext C is the output of the final round: $C = X_{10}$.



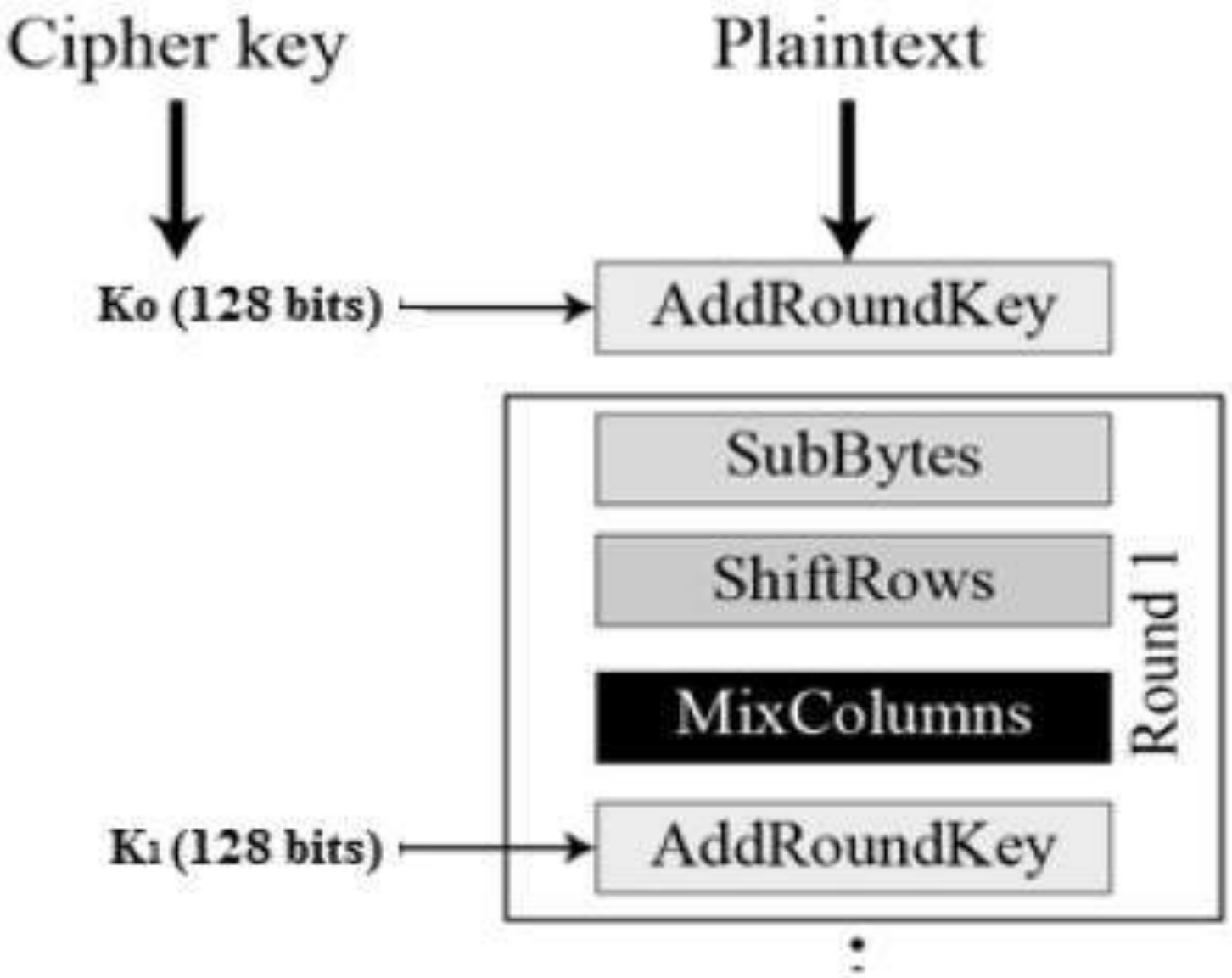



| R | Key size |
|---|---|
| 10 | 128 |
| 12 | 192 |
| 14 | 256 |

Relationship between number of rounds(R) and cipher key size

# AES Rounds

- Each round is built from four basic steps:
  1. **SubBytes step**: an S-box substitution step
  2. **ShiftRows step**: a permutation step
  3. **MixColumns step**: a matrix multiplication step
  4. **AddRoundKey step**: an XOR step with a **round key** derived from the 128-bit encryption key

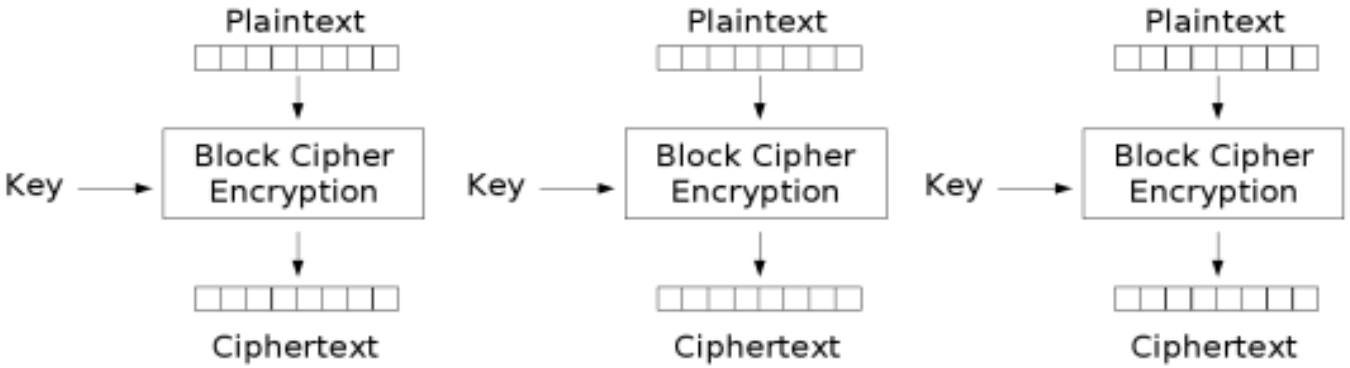

Cipher key
Plaintext
Ko (128 bits)
AddRoundKey
SubBytes
ShiftRows
MixColumns
Round 1
K1 (128 bits)
AddRoundKey

# Block Cipher Modes

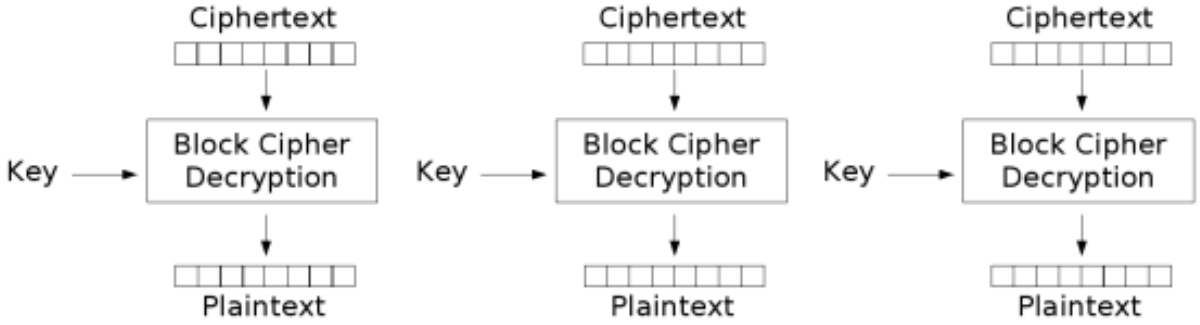
- A block cipher mode describes the way a block cipher encrypts and decrypts a sequence of message blocks.
- Different modes result in different properties being achieved. This can add to the security of the underlying block cypher.

# Electronic Code Book (ECB) Mode

- Electronic Code Book (ECB) Mode (is the simplest):
  - User takes the first block and encrypt it with the key to produce the first block of ciphertext, follow the same procedure for all following blocks.
    - Block P[i] encrypted into ciphertext block C[i] = $E_K(P[i])$
    - Block C[i] decrypted into plaintext block M[i] = $D_K(C[i])$
  - ECB mode is deterministic.
  - For given key, we can create a codebook of ciphertexts for all possible plaintext blocks. Then encryption could be as simple as looking up the codebook for ciphertext.



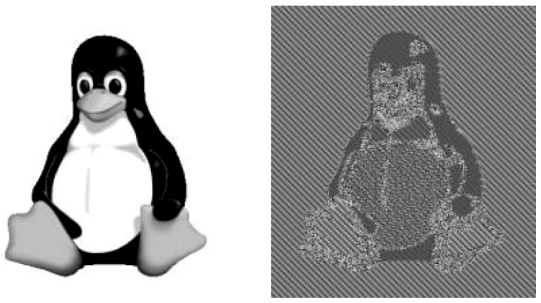

Electronic Codebook (ECB) mode encryption




Electronic Codebook (ECB) mode decryption

# Strength & Weakness of ECB

- Strengths:
  - Is very simple
  - Allows for parallel encryptions of the blocks of a plaintext
  - Can tolerate the loss or damage of a block
- Weakness:
  - Documents and images are not suitable for ECB encryption since patterns in the plaintext are repeated in the ciphertext



(a) (b)

**Figure 8.6:** How ECB mode can leave identifiable patterns in a sequence of blocks: (a) An image of Tux the penguin, the Linux mascot. (b) An encryption of the Tux image using ECB mode. (The image in (a) is by Larry Ewing, lewing@isc.tamu.edu, using The Gimp; the image in (b) is by Dr. Juzam. Both are used with permission via attribution.)

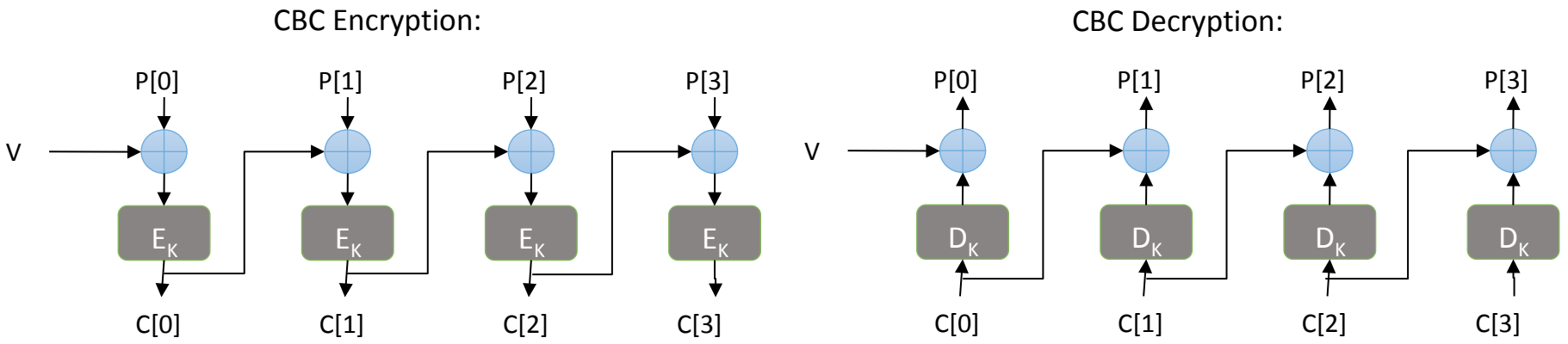
Source: Introduction to Computer Security by Goodrich & Tamassia

# Cipher Block Chaining (CBC) Mode

- This mode provides message dependence for generating ciphertext—the system is non-deterministic

# Cipher Block Chaining (CBC) Mode

- In Cipher Block Chaining (CBC) Mode
  - The previous ciphertext block is combined with the current plaintext block $C[i] = E_K(C[i-1] \oplus P[i])$
  - C[-1] = V, a random block separately transmitted encrypted (known as the initialization vector)
  - Decryption: $P[i] = C[i-1] \oplus D_K(C[i])$



# Strength and Weakness of CBC

- Strengths:
  - Doesn't show patterns in the plaintext
  - Is the most common mode
  - Is fast and relatively simple
- Weaknesses:
  - CBC requires the reliable transmission of all the blocks sequentially
  - CBC is not suitable for applications that allow packet losses (e.g., music and video streaming)

# Java AES Encryption Examples

- Source

  http://java.sun.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html

- Generate an AES key

```
KeyGenerator keygen = KeyGenerator.getInstance("AES");
SecretKey aesKey = keygen.generateKey();
```

- Create a cipher object for AES in ECB mode and PKCS5 padding

```
Cipher aesCipher;
aesCipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
```

- Encrypt

```
aesCipher.init(Cipher.ENCRYPT_MODE, aesKey);
byte[] plaintext = "My secret message".getBytes();
byte[] ciphertext = aesCipher.doFinal(plaintext);
```

- Decrypt

```
aesCipher.init(Cipher.DECRYPT_MODE, aesKey);
byte[] plaintext1 = aesCipher.doFinal(ciphertext);
```

# Public-Key Cryptography

- Utilizes the public key-private key pair.

- Modern cryptography often view blocks of bits as large numbers represented in binary.
  - Need cryptographic tools to operate on large numbers

- When we operate on blocks of bits as large numbers, we need to make sure all operations result in output values that can be represented using the same number of bits as the input value:
  - Perform all arithmetic modulo the same number n—return the remainder of a division of the result with n.

# RSA Cryptosystem

- RSA is a public-key cryptosystem
- It was invented by Ronal Rivest, Adi Shamir, and Leonard Adleman.
- In this cryptosystem:
  - Plaintext and ciphertext message blocks are treated as large numbers—represented using thousands of bits.
  - Encryption and decryption are done using modular exponentiation. The correctness is based on Euler's Theorem.

# RSA Key Pair Generation-I

- Generate the RSA modulus (n)
  - Select two large primes, p and q
  - Calculate n = p*q, for strong unbreakable encryption, let n be a large number, typically a minimum of 512 bits.
- Find derived number (e)
  - Number e must be greater than 1 and less than (p-1) (q-1)
  - There must be no common factor for e and (p-1)(q-1) except for 1, i.e., they are relatively prime.

# RSA Key Pair Generation-II

- Form the public key
  - The pair of numbers (n, e) form the RSA public key
  - Difficulty in factoring a large prime number ensures that attackers cannot find in finite time the value of p and q.
- Generate the private key
  - Private key d is calculated from p, q, and e. For given n and e, there is a unique number d.
  - Number d is the inverse of e mod (p-1)(q-1), mathematically: ed = 1 mod (p-1)(q-1).

# Optional Math for RSA

# Basics of Prime Numbers

- Prime number $p$:
  - $p$ is an integer
  - $p \geq 2$
  - The only divisors of $p$ are $1$ and $p$
- Examples
  - $2, 7, 19$ are primes
  - $-3, 0, 1, 6$ are not primes
- Prime decomposition of a positive integer $n$:

$$n = p_1^{e_1} \times \ldots \times p_k^{e_k}$$

- Example:
  - $200 = 2^3 \times 5^2$

## Fundamental Theorem of Arithmetic

The prime decomposition of a positive integer is unique

# Greatest Common Divisor

- The greatest common divisor (GCD) of two positive integers $a$ and $b$, denoted $\gcd(a, b)$, is the largest positive integer that divides both $a$ and $b$
- The above definition is extended to arbitrary integers
- Examples:

$\gcd(18, 30) = 6$ $\quad\quad$ $\gcd(0, 20) = 20$

$\gcd(-21, 49) = 7$

- Two integers a and b are said to be relatively prime if

$$\gcd(a, b) = 1$$

- Example:
  - Integers 15 and 28 are relatively prime

# Modular Arithmetic

- Modulo operator for a positive integer $n$

$$r = a \bmod n$$

equivalent to

$$a = r + kn$$

and

$$r = a - \lfloor a/n \rfloor \, n$$

- Example:

$29 \bmod 13 = 3$ $\quad$ $13 \bmod 13 = 0$ $\quad$ $-1 \bmod 13 = 12$

$29 = 3 + 2 \times 13$ $\quad$ $13 = 0 + 1 \times 13$ $\quad$ $12 = -1 + 1 \times 13$

- Modulo and GCD:

$$\gcd(a, b) = \gcd(b, a \bmod b)$$

- Example:

$\gcd(21, 12) = 3$ $\quad$ $\gcd(12, 21 \bmod 12) = \gcd(12, 9) = 3$

# Euclid's GCD Algorithm

- Euclid's algorithm for computing the GCD repeatedly applies the formula

$$\gcd(\boldsymbol{a}, \boldsymbol{b}) = \gcd(\boldsymbol{b}, \boldsymbol{a} \bmod \boldsymbol{b})$$

- Example
  - $\gcd(412, 260) = 4$

| a | 412 | 260 | 152 | 108 | 44 | 20 | 4 |
|---|---|---|---|---|---|---|---|
| b | 260 | 152 | 108 | 44 | 20 | 4 | 0 |

# Analysis

- Let $a_i$ and $b_i$ be the arguments of the $i$-th recursive call of algorithm ***EuclidGCD***
- We have

$$a_{i+2} = b_{i+1} = a_i \bmod a_{i+1} < a_{i+1}$$

- Sequence $a_1, a_2, \ldots, a_n$ decreases exponentially, namely

$$a_{i+2} \leq \tfrac{1}{2} a_i \text{ for } i > 1$$

Case 1 $a_{i+1} \leq \tfrac{1}{2} a_i$ $\quad a_{i+2} < a_{i+1} \leq \tfrac{1}{2} a_i$

Case 2 $a_{i+1} > \tfrac{1}{2} a_i$ $\quad a_{i+2} = a_i \bmod a_{i+1} = a_i - a_{i+1} \leq \tfrac{1}{2} a_i$

- Thus, the maximum number of recursive calls of algorithm ***EuclidGCD***$(a, b)$ is

$$1 + 2 \log \max(a, b)$$

- Algorithm ***EuclidGCD***$(a, b)$ executes $O(\log \max(a, b))$ arithmetic operations
- The running time can also be expressed as $O(\log \min(a, b))$

# Multiplicative Inverse-I

- The residues modulo a positive integer $n$ are the set

$$Z_n = \{0, 1, 2, \ldots, (n-1)\}$$

- Let $x$ and $y$ be two elements of $Z_n$ such that

$$xy \bmod n = 1$$

We say that $y$ is the multiplicative inverse of $x$ in $Z_n$ and we write $y = x^{-1}$

- Example:
  - Multiplicative inverses of the residues modulo 11

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $x^{-1}$ | | 1 | 6 | 4 | 3 | 9 | 2 | 8 | 7 | 5 | 10 |

# Multiplicative Inverse-II

Theorem

An element $x$ of $Z_n$ has a multiplicative inverse if and only if $x$ and $n$ are relatively prime

- Example
  - The elements of $Z_{10}$ with a multiplicative inverse are $1, 3, 7, 9$

Corollary

If is $p$ is prime, every nonzero residue in $Z_p$ has a multiplicative inverse

Theorem

A variation of Euclid's GCD algorithm computes the multiplicative inverse of an element $x$ of $Z_n$ or determines that it does not exist

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| $x^{-1}$ | | 1 | | 7 | | | | 3 | | 9 |

# Fermat's Little Theorem

## Theorem

Let $p$ be a prime. For each nonzero residue $x$ of $Z_p$, we have $x^{p-1} \bmod p = 1$

- Example ($p = 5$):

$1^4 \bmod 5 = 1$ $\quad$ $2^4 \bmod 5 = 16 \bmod 5 = 1$

$3^4 \bmod 5 = 81 \bmod 5 = 1$ $\quad$ $4^4 \bmod 5 = 256 \bmod 5 = 1$

## Corollary

Let $p$ be a prime. For each nonzero residue $x$ of $Z_p$, the multiplicative inverse of $x$ is $x^{p-2} \bmod p$

Proof

$$x(x^{p-2} \bmod p) \bmod p = xx^{p-2} \bmod p = x^{p-1} \bmod p = 1$$

# Euler's Theorem

- The multiplicative group for $Z_n$, denoted with $Z^*_n$, is the subset of elements of $Z_n$ relatively prime with $n$
- The totient function of $n$, denoted with $\phi(n)$, is the size of $Z^*_n$
- Example

$$Z^*_{10} = \{ 1, 3, 7, 9 \} \qquad \phi(10) = 4$$

- If $p$ is prime, we have

$$Z^*_p = \{1, 2, \ldots, (p-1)\} \qquad \phi(p) = p - 1$$

Euler's Theorem

For each element $x$ of $Z^*_n$, we have $x^{\phi(n)} \bmod n = 1$

- Example ($n = 10$)

$$3^{\phi(10)} \bmod 10 = 3^4 \bmod 10 = 81 \bmod 10 = 1$$
$$7^{\phi(10)} \bmod 10 = 7^4 \bmod 10 = 2401 \bmod 10 = 1$$
$$9^{\phi(10)} \bmod 10 = 9^4 \bmod 10 = 6561 \bmod 10 = 1$$

# RSA Specifics

- Setup:
  - $n = pq$, with $p$ and $q$ primes
  - $e$ relatively prime to $\phi(n) = (p - 1)(q - 1)$
  - $d$ inverse of $e$ in $Z_{\phi(n)}$
- Keys:
  - Public key: $K_E = (n, e)$
  - Private key: $K_D = d$
- Encryption:
  - Plaintext $M$ in $Z_n$
  - $C = M^e \bmod n$
- Decryption:
  - $M = C^d \bmod n$

- Example
  - Setup:
    - $p = 7,\ q = 17$
    - $n = 7 \cdot 17 = 119$
    - $\phi(n) = 6 \cdot 16 = 96$
    - $e = 5$
    - $d = 77$
  - Keys:
    - public key: $(119, 5)$
    - private key: $77$
  - Encryption:
    - $M = 19$
    - $C = 19^5 \bmod 119 = 66$
  - Decryption:
    - $C = 66^{77} \bmod 119 = 19$

# RSA Example

- Setup:
  - $p = 5, q = 11$
  - $n = 5 \cdot 11 = 55$
  - $\phi(n) = 4 \cdot 10 = 40$
  - $e = 3$
  - $d = 27$ $(3 \cdot 27 = 81 = 2 \cdot 40 + 1)$

- Encryption
  - $C = M^3 \bmod 55$
- Decryption
  - $M = C^{27} \bmod 55$

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *M* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| *C* | 1 | 8 | 27 | 9 | 15 | 51 | 13 | 17 | 14 | 10 | 11 | 23 | 52 | 49 | 20 | 26 | 18 | 2 |
| *M* | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 |
| *C* | 39 | 25 | 21 | 33 | 12 | 19 | 5 | 31 | 48 | 7 | 24 | 50 | 36 | 43 | 22 | 34 | 30 | 16 |
| *M* | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 |
| *C* | 53 | 37 | 29 | 35 | 6 | 3 | 32 | 44 | 45 | 41 | 38 | 42 | 4 | 40 | 46 | 28 | 47 | 54 |

It is difficult to find value of d given e and n. Without knowledge of p and q, it is difficult to factor n.

# Security of RSA

- Security of RSA based on difficulty of factoring
  - Widely believed
  - Best known algorithm takes exponential time
- RSA Security factoring challenge (discontinued)
- In 1999, 512-bit challenge factored in 4 months using 35.7 CPU-years
  - 160 175-400 MHz SGI and Sun
  - 8 250 MHz SGI Origin
  - 120 300-450 MHz Pentium II
  - 4 500 MHz Digital/Compaq
- In 2005, a team of researchers factored the RSA-640 challenge number using 30 2.2GHz CPU years
- In 2004, the prize for factoring RSA-2048 was $200,000
- Current practice is 2,048-bit keys
- Estimated resources needed to factor a number within one year

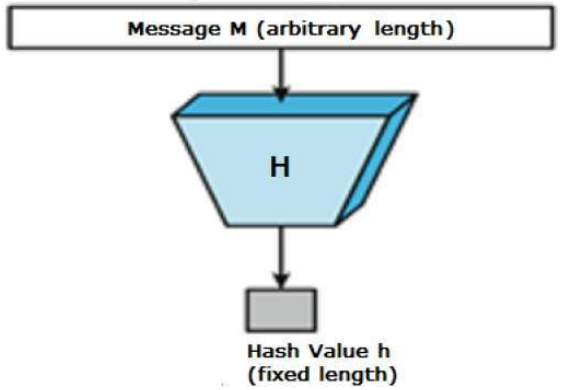
| Length (bits) | PCs | Memory |
|---|---|---|
| 430 | 1 | 128MB |
| 760 | 215,000 | 4GB |
| 1,020 | $342 \times 10^6$ | 170GB |
| 1,620 | $1.6 \times 10^{15}$ | 120TB |

# RSA Algorithms

- The implementation of the RSA cryptosystem requires various algorithms
- Overall
  - Representation of integers of arbitrarily large size and arithmetic operations on them
- Encryption
  - Modular power
- Decryption
  - Modular power
- Setup
  - Generation of random numbers with a given number of bits (to generate candidates $p$ and $q$)
  - Primality testing (to check that candidates $p$ and $q$ are prime)
  - Computation of the GCD (to verify that $e$ and $\phi(n)$ are relatively prime)
  - Computation of the multiplicative inverse (to compute $d$ from $e$)

# Cryptography Hash Functions

- A hash function is a mathematical function that converts a numerical input value into another compressed numerical value.
- A hash function h maps a plaintext x to a fixed-length value x = h(P) called hash value or digest of P
- The input to the hash function is of arbitrary length but output is of fixed length.



# Hash Function Basics

- Hash function with n bit output is referred to as an **n-bit hash function**. Popular hash functions generate values between 160 and 512 bits.
- A collision is a pair of plaintexts P and Q that map to the same hash value, h(P) = h(Q). Collisions are unavoidable
- For efficiency, the computation of the hash function should take time proportional to the length of the input plaintext

# Properties of Hash Functions-I

- **Pre-Image Resistance**
  - This property means that it should be computationally hard to reverse a hash function.
  - In other words, if a hash function h produced a hash value z, then it should be a difficult process to find any input value x that hashes to z.
  - This property protects against an attacker who only has a hash value and is trying to find the input.

# Properties of Hash Functions-II

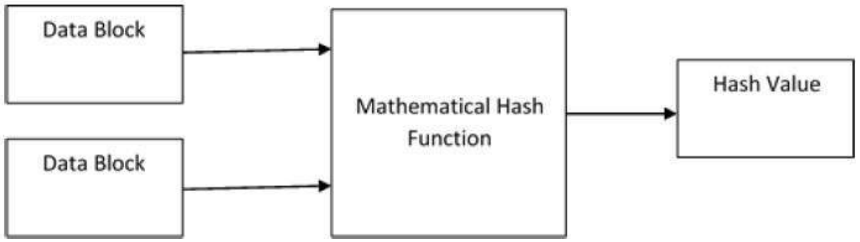
- **Second Pre-Image Resistance**
  - This property means given an input and its hash, it should be hard to find a different input with the same hash.
  - In other words, if a hash function h for an input x produces hash value h(x), then it should be difficult to find any other input value y such that h(y) = h(x).
  - This property of hash function protects against an attacker who has an input value and its hash, and wants to substitute different value as legitimate value in place of original input value.

# Properties of Hash Functions-III

- Collision Resistance
  - For a hash function h, it is hard to find any two different inputs x and y such that h(x) = h(y).
  - Since, hash function is compressing function with fixed hash length, it is impossible for a hash function not to have collisions. This property of collision free only confirms that these collisions should be hard to find.
  - This property makes it very difficult for an attacker to find two input values with the same hash.
  - Also, if a hash function is collision-resistant **then it is second pre-image resistant.**

# Hash Algorithms and Hash Functions

- A hash function generated a hash code by operating on two blocks of fixed-size data to create a hash code.




- A hash algorithm defines the process of using the hash function—how the message will be broken up and how results from previous message blocks are chained together.

# Hash Functions-Message Digest (MD)

- MD5 was most popular and widely used for many years.
  - The MD family comprises of hash functions MD2, MD4, MD5 and MD6. It was adopted as Internet Standard RFC 1321. It is a 128-bit hash function.
  - MD5 digests have been widely used in the software world to provide assurance about integrity of transferred file. For example, file servers often provide a pre-computed MD5 checksum for the files, so that a user can compare the checksum of the downloaded file to it.
  - In 2004, collisions were found in MD5. An analytical attack was reported to be successful only in an hour by using computer cluster. This collision attack resulted in compromised MD5 and hence it is no longer recommended for use.

# Secure Hash Algorithm (SHA)-I

- Developed by NSA and approved as a federal standard by NIST
- SHA-0 and SHA-1 (1993)
  - 160-bits
  - Considered insecure
  - Still found in legacy applications
  - Vulnerabilities less severe than those of MD5
- SHA-2 family (2002)
  - 256 bits (SHA-256) or 512 bits (SHA-512)
  - Still considered secure despite published attack techniques
- Public competition for SHA-3 announced in 2007

# Secure Hash Algorithm (SHA)-II

- The original version is SHA-0, a 160-bit hash function, was published by the National Institute of Standards and Technology (NIST) in 1993. It had few weaknesses and did not become very popular. Later in 1995, SHA-1 was designed to correct alleged weaknesses of SHA-0.

- SHA-1 is the most widely used of the existing SHA hash functions. It is employed in several widely used applications and protocols including Secure Socket Layer (SSL) security.

- In 2005, a method was found for uncovering collisions for SHA-1 within practical time frame making long-term employability of SHA-1 doubtful.

# Secure Hash Algorithm (SHA)-II

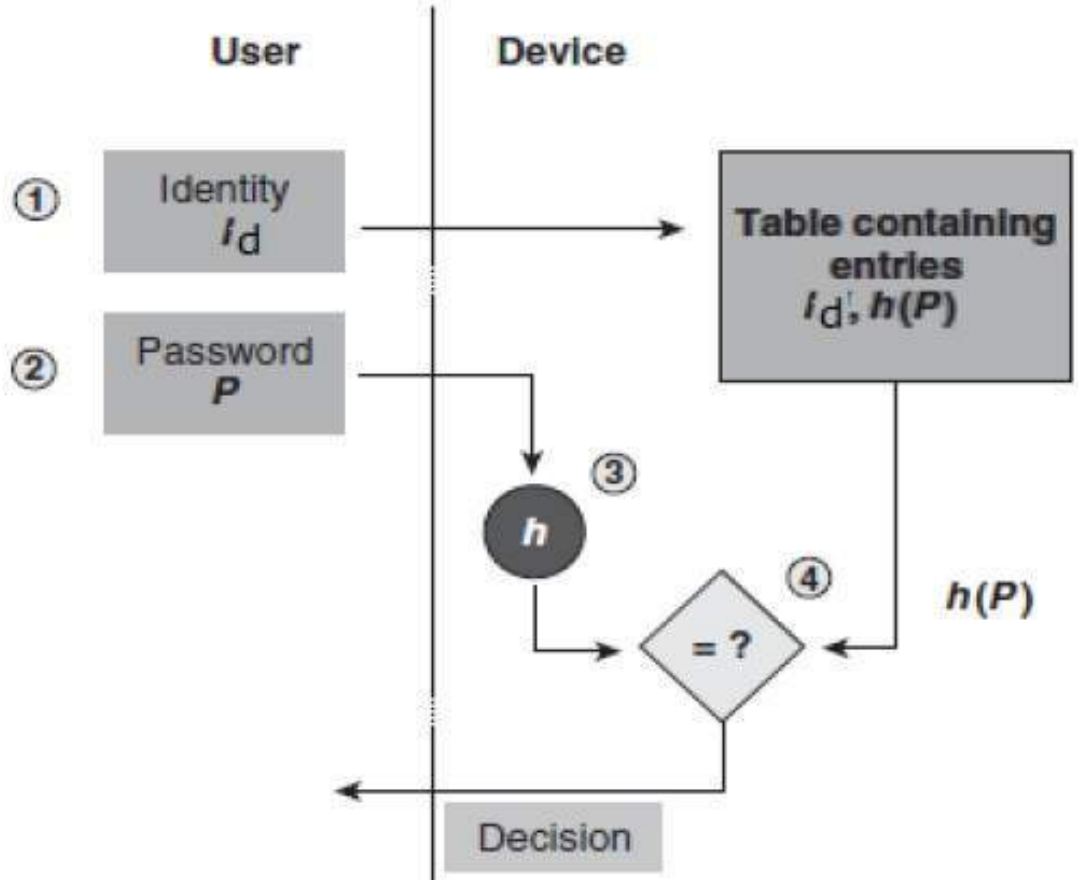
- SHA-2 family has four further SHA variants, SHA-224, SHA-256, SHA-384, and SHA-512 depending up on number of bits in their hash value. No successful attacks have yet been reported on SHA-2 hash function.

- Though SHA-2 is a strong hash function. Though significantly different, its basic design is still follows design of SHA-1. Hence, NIST called for new competitive hash function designs.

- In October 2012, the NIST chose the Keccak algorithm as the new SHA-3 standard. Keccak offers many benefits, such as efficient performance and good resistance for attacks.

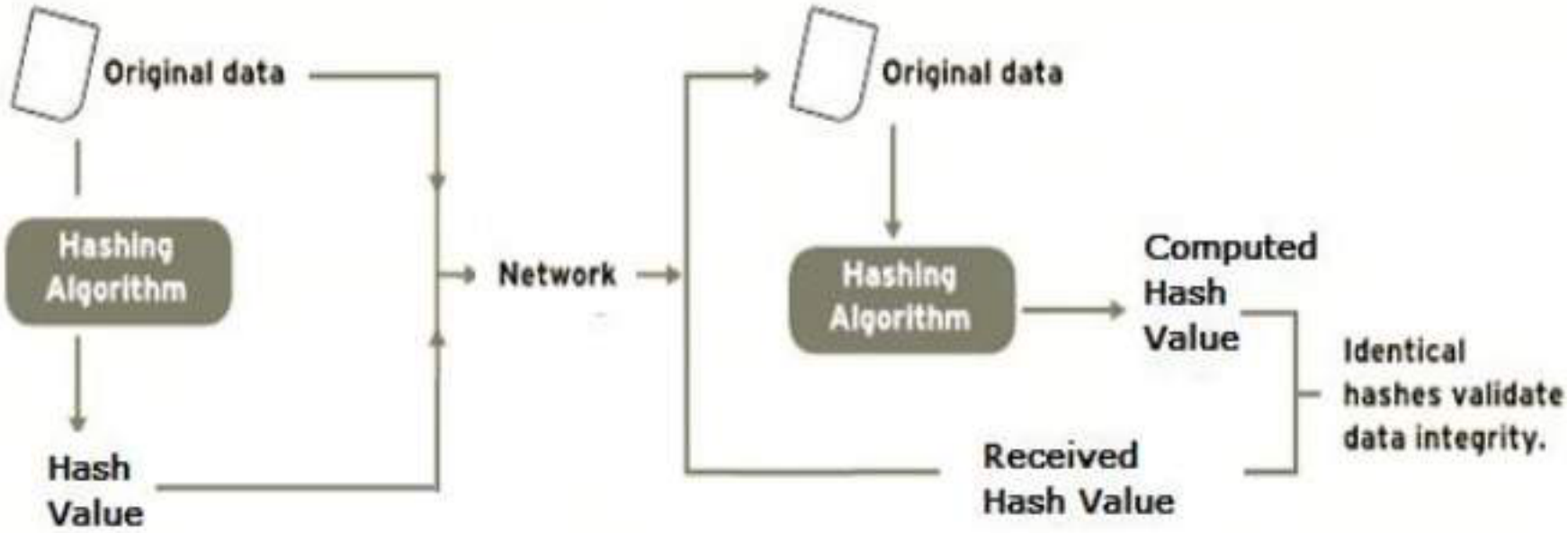
# Applications of Hash Functions-I

- Password Storage



# Applications of Hash Functions-II

- Data Integrity Check
  - Most common use of hash functions
  - Generate checksum to assure the correctness of the data



# Birthday Attack-I

- The brute-force birthday attack aims at finding a collision for a hash function h
  - Randomly generate a sequence of plaintexts $X_1$, $X_2$, $X_3$,…
  - For each $X_i$ compute $y_i = h(X_i)$ and test whether $y_i = y_j$ for some $j < i$
  - Stop as soon as a collision has been found
- If there are m possible hash values, the probability that the i-th plaintext does not collide with any of the previous i −1 plaintexts is $1 - (i - 1)/m$
- The probability $F_k$ that the attack fails (no collisions) after k plaintexts is

$$F_k = (1 - 1/m)\,(1 - 2/m)\,(1 - 3/m) \ldots (1 - (k - 1)/m)$$

# Birthday Attack

- Using the standard approximation $1 - x \approx e^{-x}$

$$F_k \approx e^{-(1/m + 2/m + 3/m + \ldots + (k-1)/m)} = e^{-k(k-1)/2m}$$

- The attack succeeds/fails with probability ½ when $F_k = ½$ , that is,

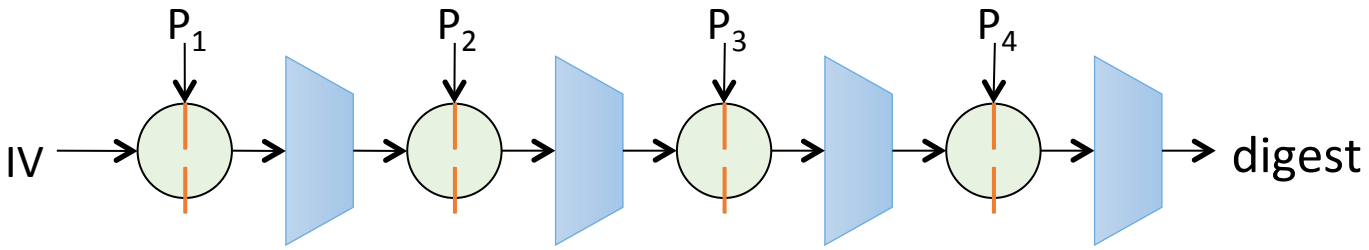
$$e^{-k(k-1)/2m} = ½$$

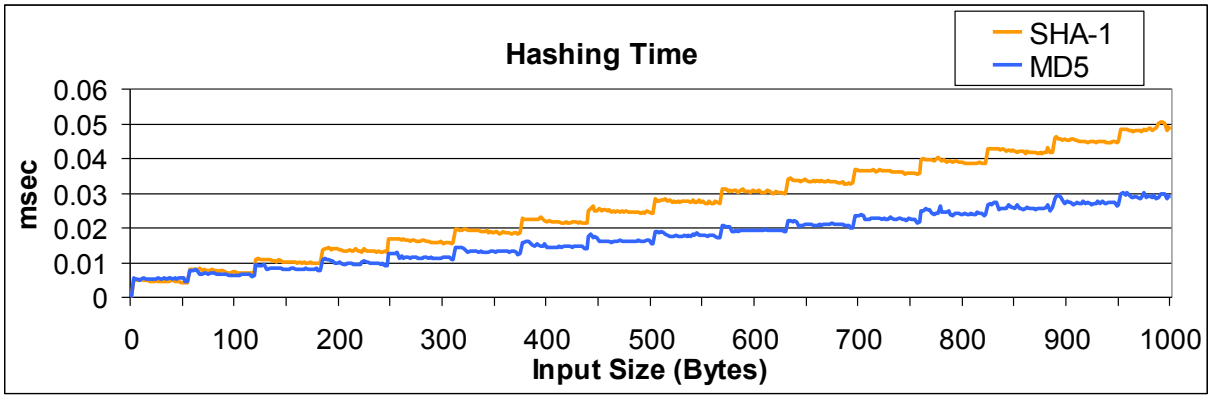
$$k \approx 1.17\, m^{½}$$

- We conclude that a hash function with b-bit values provides about b/2 bits of security

# Iterated Hash Function

- A compression function works on input values of fixed length
- An iterated hash function extends a compression function to inputs of arbitrary length
  - padding, initialization vector, and chain of compression functions
  - inherits collision resistance of compression function
- MD5 and SHA are iterated hash functions


P1
P2
P3
P4
IV
digest


Hashing Time
SHA-1
MD5
msec
0.06
0.05
0.04
0.03
0.02
0.01
0
0
100
200
300
400
500
600
700
800
900
1000
Input Size (Bytes)