
# Security Mechanisms and Architectures, Access Control, Cryptography

# Security Objectives/Goals: Information System

- CIA triad

- Confidentiality (or secrecy)
  - unauthorized users cannot read information
- Integrity
  - unauthorized users cannot alter information
- Availability
  - authorized users can always access information

- Related properties:
  - Availability: Robustness (critical systems …)
  - Confidentiality/Integrity : Privacy Protection

# Security Services: processes/communication

- **Security services according to ITU-T X-800 (OSI context) :**
  - **Confidentiality** (prevent acquiring knowledge of some data)
    - Information (application data, network data like headers)
    - Network traffic (metadata revealing information flow between entities)
  - **Integrity**
    - Of data (preventing data modification)
    - Of program and their execution (original code, vulnerabilities)
  - **Authentication**
    - Of data / program origin (prevent the injection of fake data)
    - Of entities (prevent identity theft)
  - **Authorization / Access Control** (prevent unauthorized accesses)
  - **Non-repudiation** (prevent a posteriori denial of a transaction)
    - With proof of origin
    - With proof of delivery
- **More:**
  - traceability and auditing (forensics)
  - monitoring (real-time auditing)
  - multi-level security
  - privacy & anonymity

# How to Realize Security Objectives?

- Approaches:
  - Prevention
    - measures to stop breaches of security goals
  - Detection
    - measures to detect breaches of security goals
  - Reaction
    - measures to recover assets, repair damage, and fight (and deter) offenders
- Good prevention does not make detection & reaction superfluous
  - E.g., breaking into any house with windows is trivial; despite this absence of prevention, detection & reaction still deter burglars
- Defense in depth!!

# Security Mechanisms

- Cryptography
  - for threats related to insecure communication and storage
  - Also used for algorithmic access control
- Logical Access control and Security Architectures
  - for threats related to misbehaving users
  - Definition and defense of a perimeter
  - Data / access protection
  - Access control models (E.g., role-based access control)
- Language-based security
  - for threats related to misbehaving programs
    - typing, memory-safety, modules, packages …
    - sandboxing
  - E.g., Java, .NET/C#, Rust

# Cryptography:
# Encryption, Hash Functions, Certificates

# Encryption

- Transforms data (also called plaintexts or cleartexts) into a ciphertext
  - The ciphertext cannot be read by anybody outside those who possess a secret
  - The secret is also called an encryption or decryption key
- Encryption is a mechanism ensuring the confidentiality property
- Kerckhoffs principle (1883):
  - The security of a system must only rely on the ignorance of the key, not that of another parameter
  - No ”security by obscurity”

Enigma Machine

Augustin Kerckhoffs

# Symmetric Key Cryptography

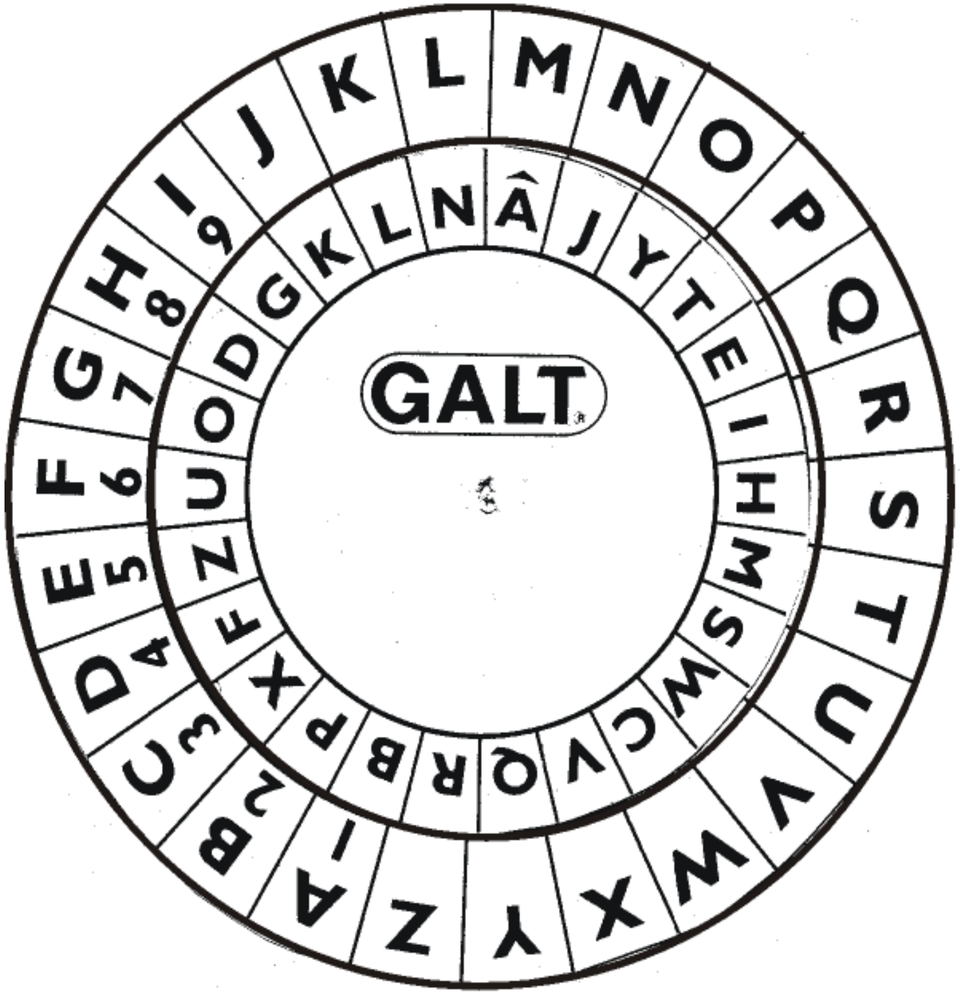

- Encryption and decryption are symmetrical and use the same key
- Main problem of symmetric key encryption: key exchange
  - The keys must be exchanged between each pair of communicating entities
  - In practice, one must also distribute secret keys that will be used without these keys being intercepted by eavesdroppers.

# Encryption - approaches

- Substitution Ciphers
  - Monoalphabetic Ciphers
    - Each letter is substituted by another chosen arbitrarily
    - Weakness: frequential analysis (frequency of occurrence of letters …)
    - Example: The alphabet is shifted (rotation) - Caesar’s cipher (3 shifts)
  - Polyalphabetic substitution Ciphers
    - Permutation of alphabetic symbols
    - Using multiple ciphering alphabets (1 permutation per symbol)
    - The key selects which alphabet is used for each letter
    - Example: Vigenère’s cipher
- Permutation Ciphers (or Transposition Ciphers)
  - The letter order is modified without changing their value
  - Permuting input symbols
  - The key identifies the permutation
- Product Ciphers
  - Combination of substitutions and permutations
  - Foundation of modern encryption

# Encryption: Monoalphabetic Substitution

simple shift can be supported by simple hardware (rotating wheel)




Source: Jacques Savoy

# Encryption: Polyalphabetic Substitution

For the « B » letter in the key, a shift of +1 is used in the alphabet.

If the plaintext contains a « R », and the key a « B », the cleartext will be « S »

If the plaintext contains a « E », and the key a « A », the cleartext will remain a « E »

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| clair = | R | E | N | A | I | S | S | A | N | C | E |
| clé = | **B** | **A** | **N** | **D** | B | A | N | D | B | A | N |
| chiffrement = | S | E | A | D | J | S | F | D | O | C | R |

Blaise de Vigenère

Source: Jacques Savoy

# Encryption : Permutation

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| r | e | n | d |
| e | z | v | o |
| u | s | d | e |
| m | a | i | n |
| a | l | u | n |
| i | v | e | r |
| s | i | t | e |

Encryption Key: «3 1 4 2»

First line : «nrde»

Ciphertext:

«nrdeveozduesimnuanleirvtsei»

Source: Jacques Savoy

# Block Ciphers

- Confusion :
  - The relationship between plaintext P and ciphertext C statistics must be overly complex to be exploited through cryptanalysis
  - Base Technique: Substitution

- Diffusion :
  - Each symbol of P and/or K must influence several symbols of C
  - The plaintext redundancy must be distributed over a ciphertext
  - Base Technique: Permutation (Transposition)

# DES

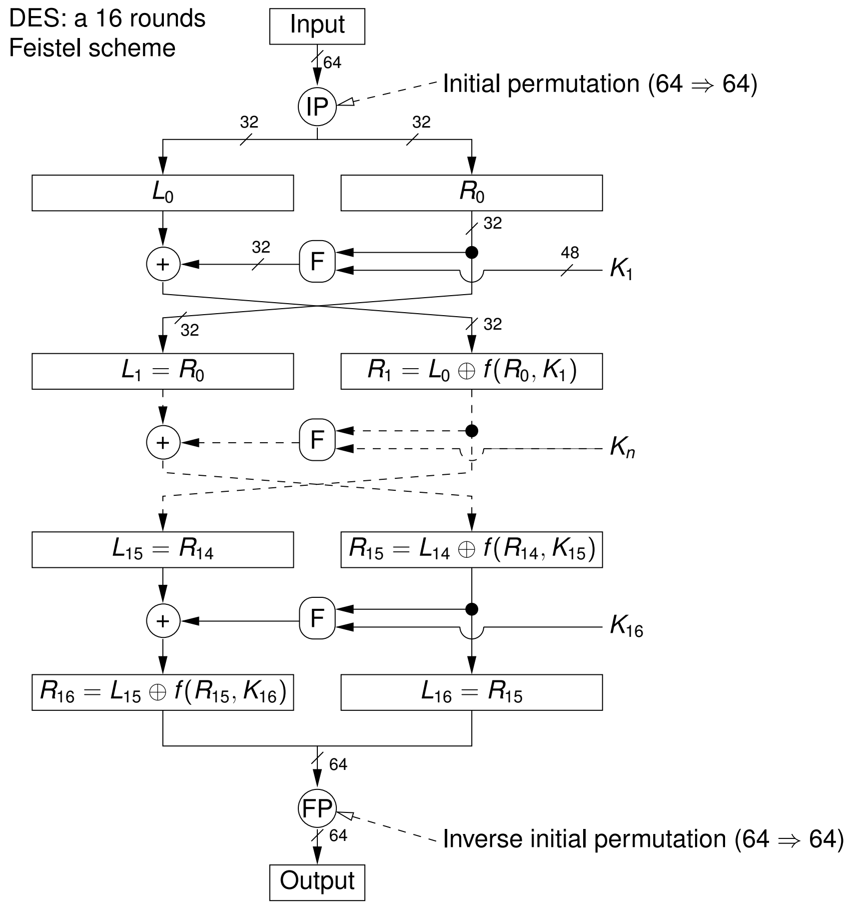
In 1973, the *National Bureau of Standards* of the USA launches a call for a cryptographic system.

In 1975, the Data Encryption Standard (DES), developed by IBM (under the name « Lucifer ») is adopted.

- 64 bit Block Cipher
- 56 bit Key (72 057 594 037 927 936 keys)

# Encryption with DES

- DES - Data Encryption Standard (1976)
  - Feistel scheme (iterative block encryption)
  - Optimized for hardware implementation
  - 56 bit key, vulnerable to exhaustive key search ($2^{56}$ possibilities)
  - A computer with 1024 processors running at 1 GHz can explore all keys in a day
  - DES is no longer secure but is now used as Triple DES (DED or EDE).

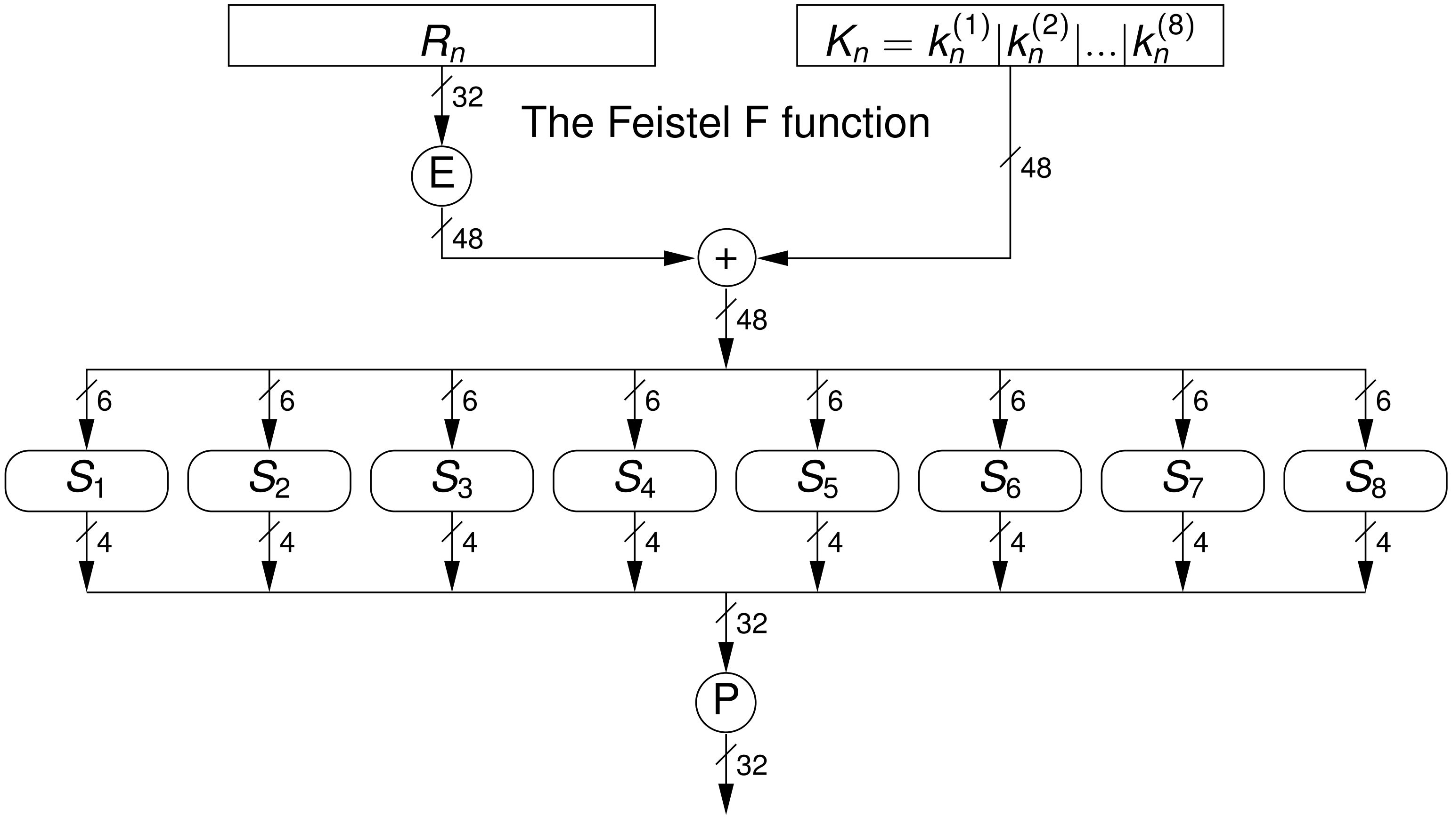



- Newer standard: AES – Advanced Encryption Standard (2000)
  - 128-256 bit keys

# The Feistel *f* function in the DES algorithm

1) 32 bit transposition towards 48 bits
2) Precomputed substitution (stored in tables called S-boxes)

S-Box design (pre-computed substitutions) was wildly discussed



# DES: Substitutions / Permutation

- S-boxes (S1-S8) – 6 bits each:

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | **0** | 14 | 4 | 13 | 1 | 2 | 15 | 11 | 8 | 3 | 10 | 6 | 12 | 5 | 9 | 0 | 7 |
| | **1** | 0 | 15 | 7 | 4 | 14 | 2 | 13 | 1 | 10 | 6 | 12 | 11 | 9 | 5 | 3 | 8 |
| | **2** | 4 | 1 | 14 | 8 | 13 | 6 | 2 | 11 | 15 | 12 | 9 | 7 | 3 | 10 | 5 | 0 |
| | **3** | 15 | 12 | 8 | 2 | 4 | 9 | 1 | 7 | 5 | 11 | 3 | 14 | 10 | 0 | 6 | 13 |

  - Line selected after bits 0|5
  - Column selected after bits 1|2|3|4
  - E.g. 101110 = line 2 (« 10 »), column 7 (« 0111 »)

- Permutation – 32 bits:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **P** | 16 | 7 | 20 | 21 | 29 | 12 | 28 | 17 |
| | 1 | 15 | 23 | 26 | 5 | 18 | 31 | 10 |
| | 2 | 8 | 24 | 14 | 32 | 27 | 3 | 9 |
| | 19 | 13 | 30 | 6 | 22 | 11 | 4 | 25 |

# Chaining Modes

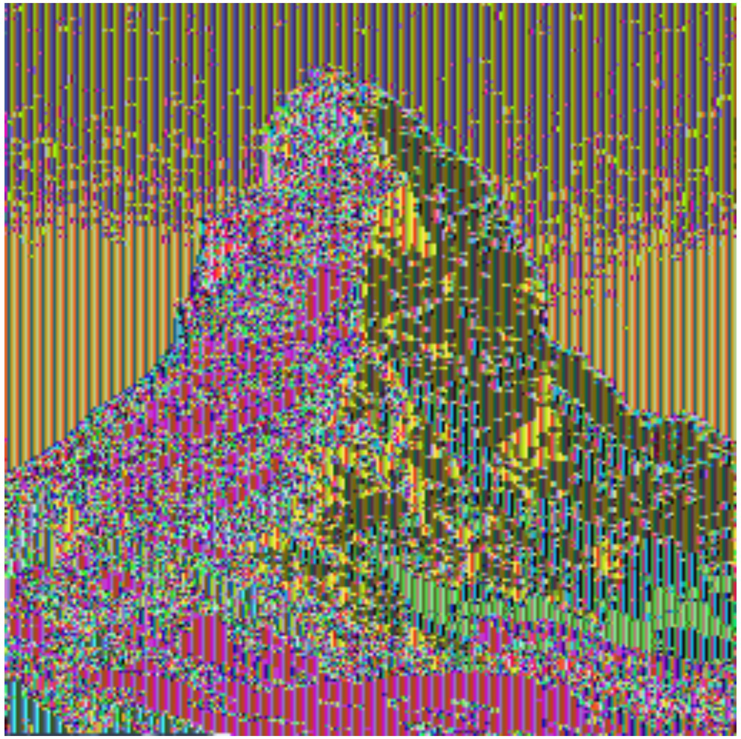
- ❒ Electronic Code Book (ECB) **– Statistical Attacks** against DES in ECB mode !

- ❒ Cipher Block Chaining (CBC)
- ❒ Cipher Feedback (CFB)
- ❒ Output Feedback (OFB)
- ❒ Counter (CTR)
- ❒ CTS, PCBC, XEX, TCB, LRW, XTS …

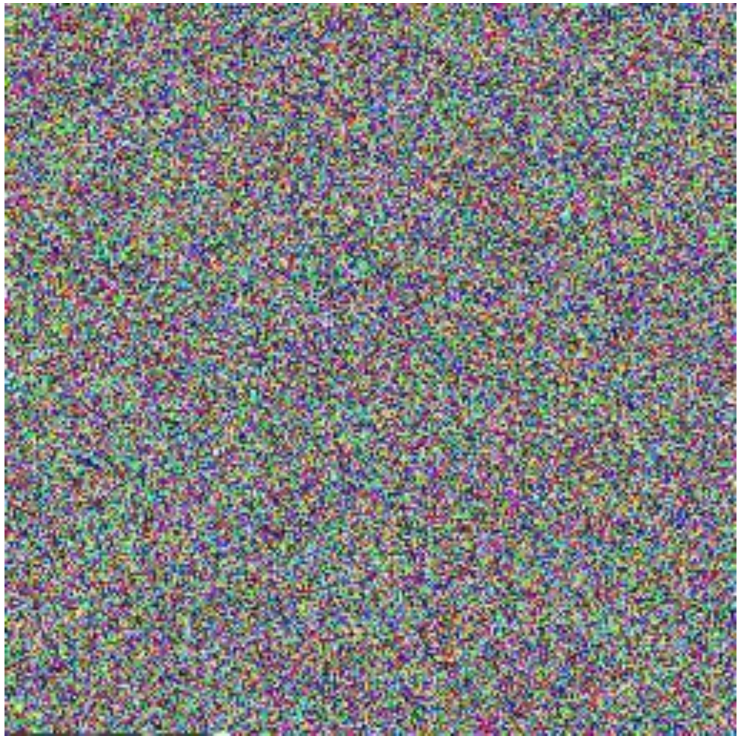
**Chaining blocks introduces problems with respect to error propagation and resynchronization …**

# Plaintext

# DES-ECB Encryption



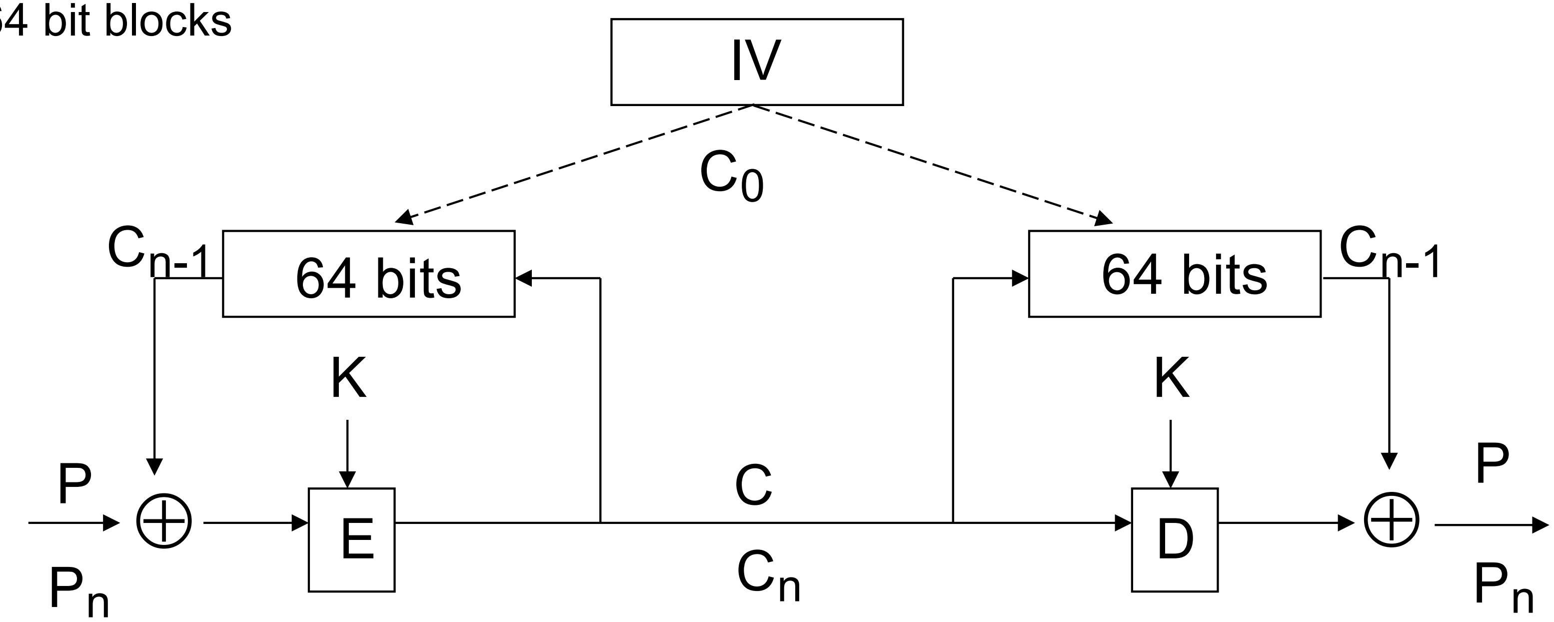
# DES-CBC Encryption



# CBC Mode

Cipher Block Chaining

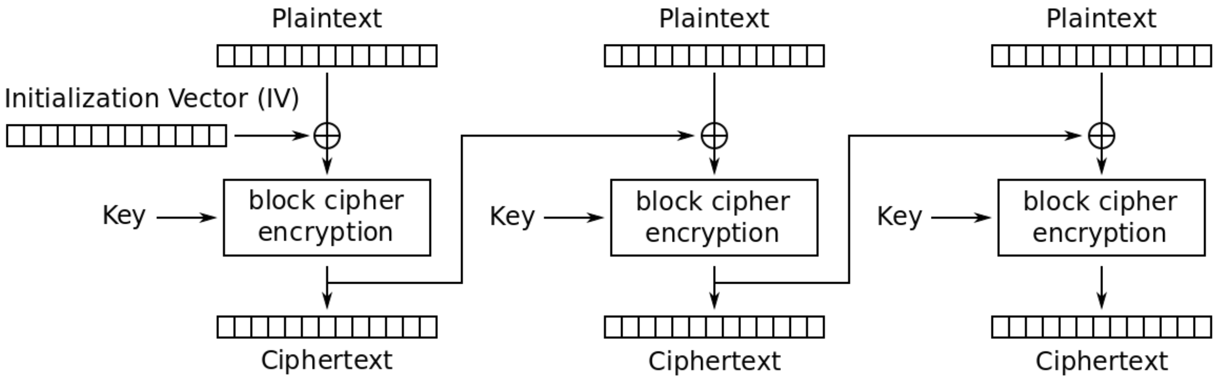
64 bit blocks



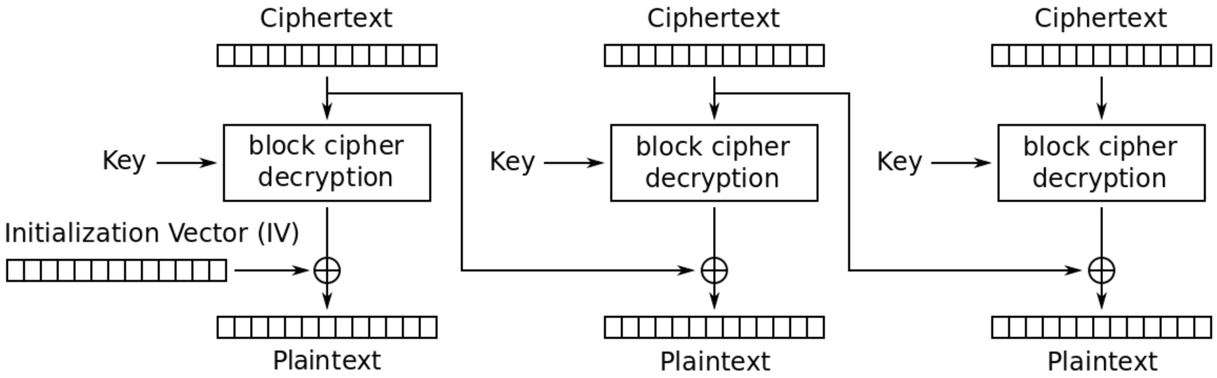

- $C_i = E_K(P_i \oplus C_{i-1})$
- $C_0 = E_K(P_0 \oplus IV)$, IV (Initialization Vector) transmitted in clear
- $P_i = D_K(C_i) \oplus C_{i-1}$

- chaining effect : $C_i$ depends on all $P_j$ with $j \leq i$
- last block in C : depends on all cleartext blocks
- converts DES into a stream cipher
- 1 encryption / decryption operation per 64 bits

# Cipher Block Chaining (CBC) Mode




Cipher Block Chaining (CBC) mode encryption

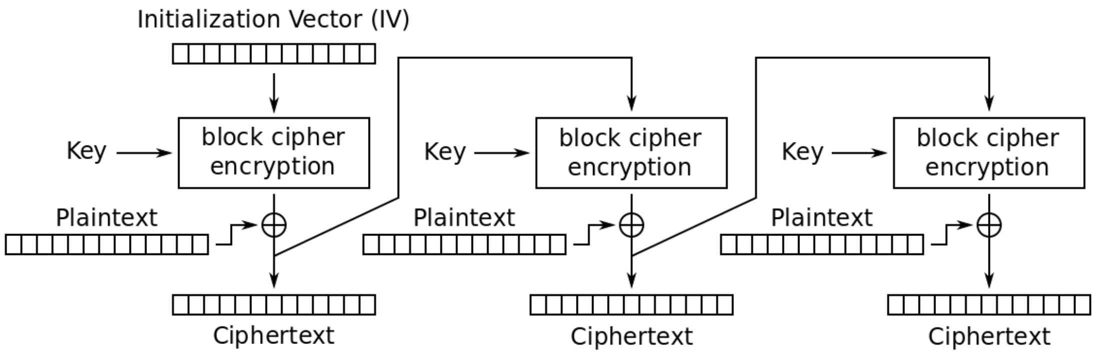



Cipher Block Chaining (CBC) mode decryption

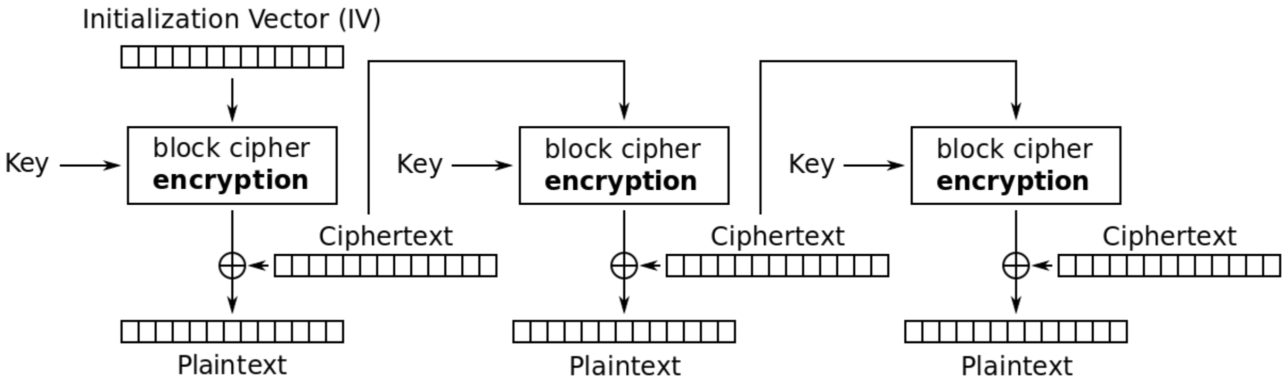
- Computation cannot be parallelized
- Requires padding if text too small (block cipher)

# Cipher Feedback (CFB) Mode

- Sort of stream cipher



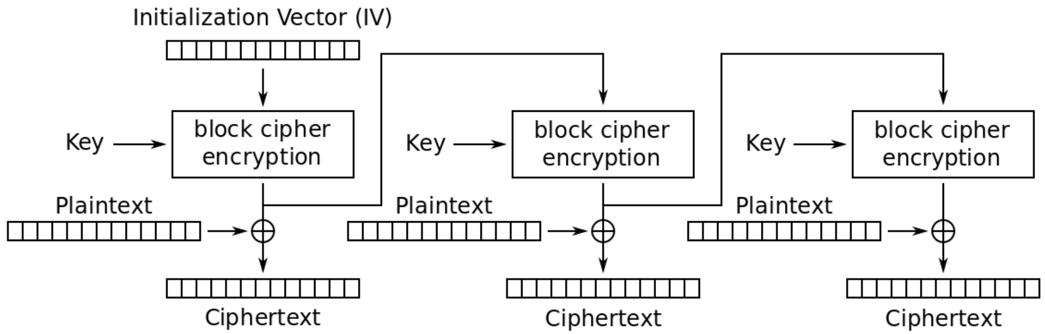

Cipher Feedback (CFB) mode encryption



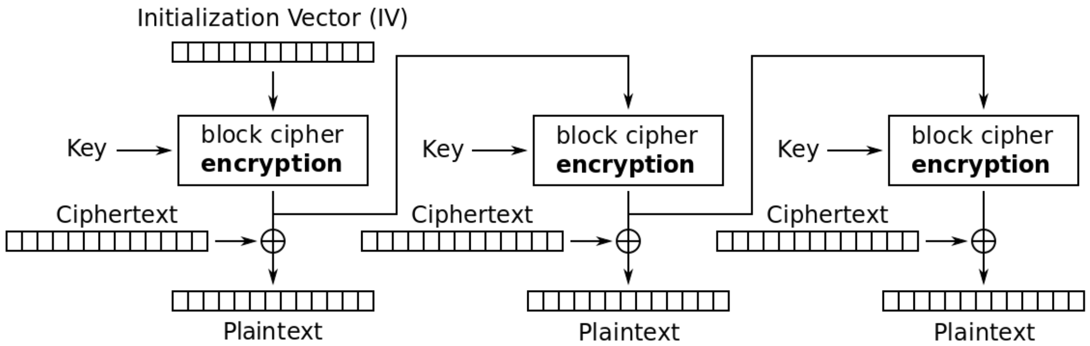

Cipher Feedback (CFB) mode decryption

# Output Feedback (OFB) Mode

- Same as CFB except keys can be precomputed

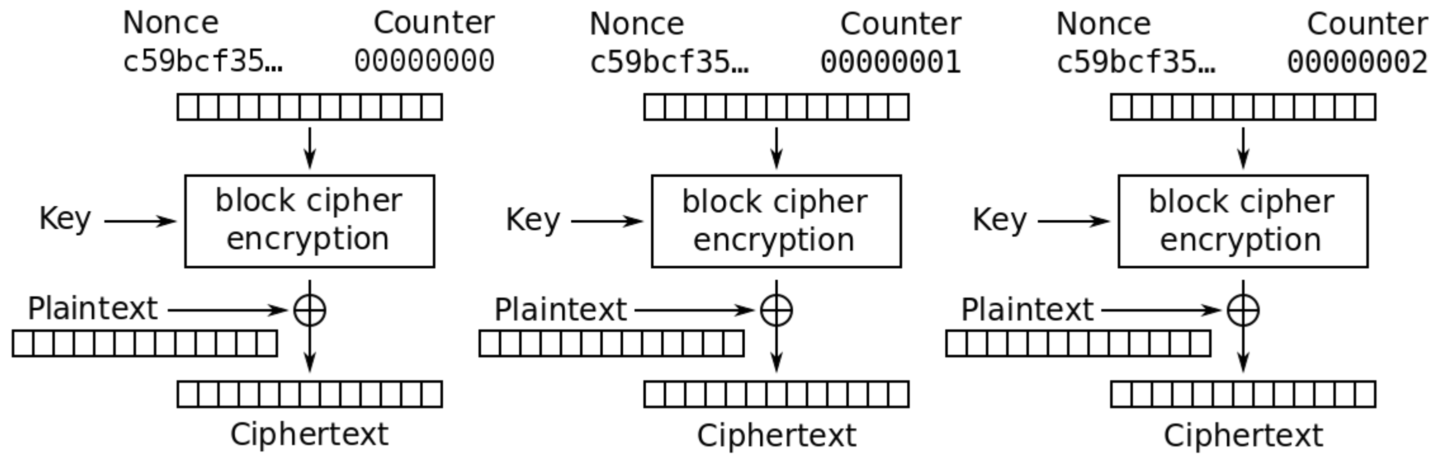



Output Feedback (OFB) mode encryption

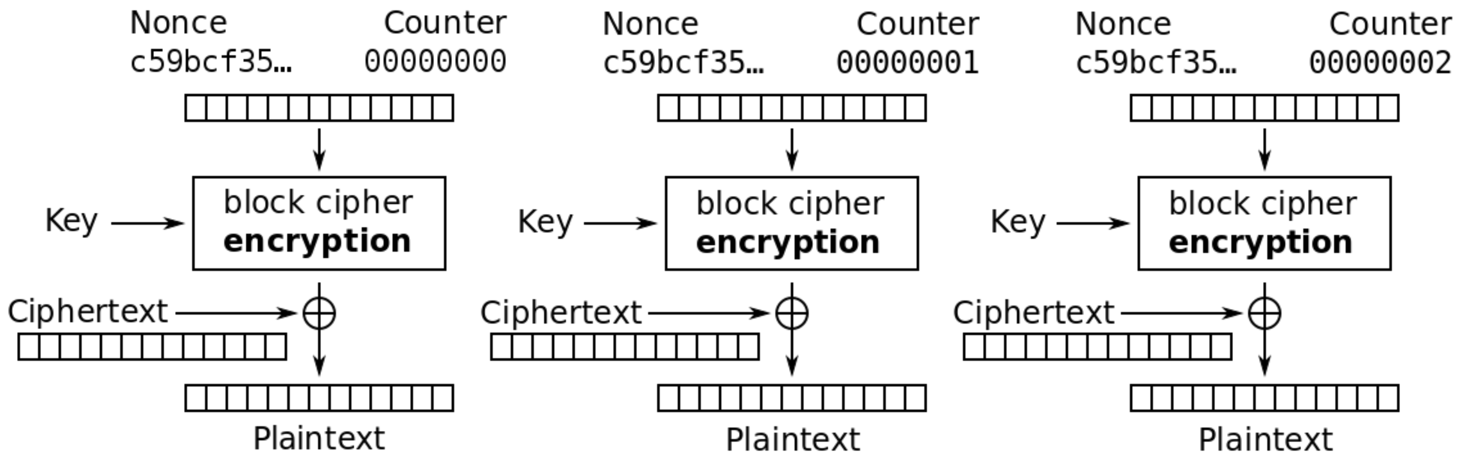



Output Feedback (OFB) mode decryption

# Counter (CTR) Mode

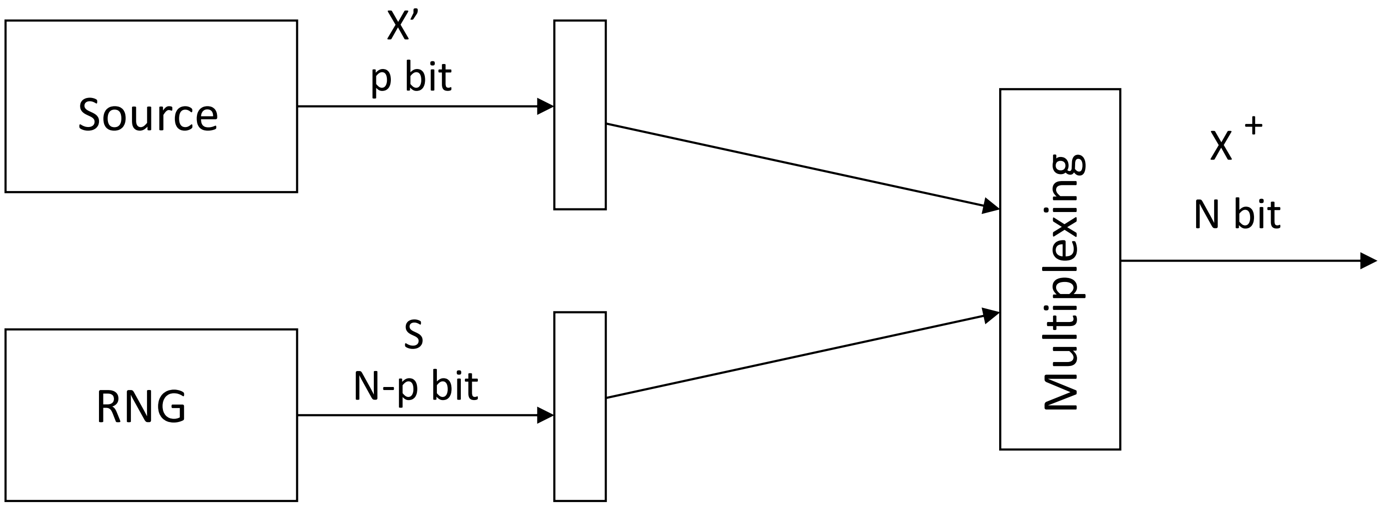



Counter (CTR) mode encryption




Counter (CTR) mode decryption

# Padding




- Two applications:
  - fighting cryptanalysis (e.g., frequency analysis)
  - Adapting cleartext to algorithms (input format size, e.g., 64 bits in DES)
- Example
  - Encryption Algorithme : 56-bit DES key, H(K) = 56 bits
  - Cleartext: 8-bit English coded in ASCII, r= 6,7 bits/character
  - Padding of every source bit with 63 random bits, p=1
- Standards:
  - PKCS#5 and PKCS#7 (Public Key Cryptography Standard), PKCS#1 - RSA-OAEP (Optimal Asymmetric Encryption Padding), zero padding for hashes (ISO/EIC 10118-1) …

Adapté d'après R. Molva

# Types of attacks

- ***Ciphertext only attacks***:
  the attacker knows C with $C=E_K(P)$
  objectives: find P and K
- ***Known cleartext attacks***:
  the attacker knows $(P_i , C_i)$ with $C_i = E_K(P_i )$
  objectives: find K
- ***Chosen cleartext attacks***:
  the attacker can obtain $C_i$ starting from a chosen $P_i$
  objectives : find K
  the attacks can be adaptive
- ***Chosen ciphertext attacks***:
  the attacker can obtain $P_i$ for a chosen $C_i$
  objectives : find K
  the attacks can also be adaptive

# Security Evaluation

- **Unconditional Security (Perfect Secrecy)** = the system is secure against an attacker with an unlimited amount of time or resources
  - so: is there enough information to compromise the system security?

- **Security through theoretical complexity** = proving that the system is secure against an adversary with polynomial capability.

- **Provable Security** = proving that compromising the system security amounts to solving a problem known as "difficult" (e.g., discrete logarithm, factoring big integers).

- **Computational Security** (practical security) = the system is secure against an attacker with a given time and resources.

# One-time pad: perfect secrecy

***Vernam Cipher***

$$E \begin{cases} P & 01001011100\ \ldots.. \\ & \oplus \\ K & 10011101010\ \ldots. \end{cases}$$

$$\downarrow$$

$$D \begin{cases} C & 11010110110\ \ldots. \\ & \oplus \\ K & 10011101010\ \ldots. \end{cases}$$

$$\downarrow$$

$$P \quad 01001011100\ \ldots.$$

Gilbert Vernam

- Additive group operation
- Implementation using exclusive or (XOR)



Adapté d'après R. Molva

# Hash Functions

- A hash function h is a function which associates to a message M of any length a message h(M) (also denoted as $\{M\}^h$) of a constant (generally short) length.
- While M can be arbitrarily large while $\{M\}^h$ has a given length

Example: file system hashcodes, error detection codes

# Cryptographic hash functions

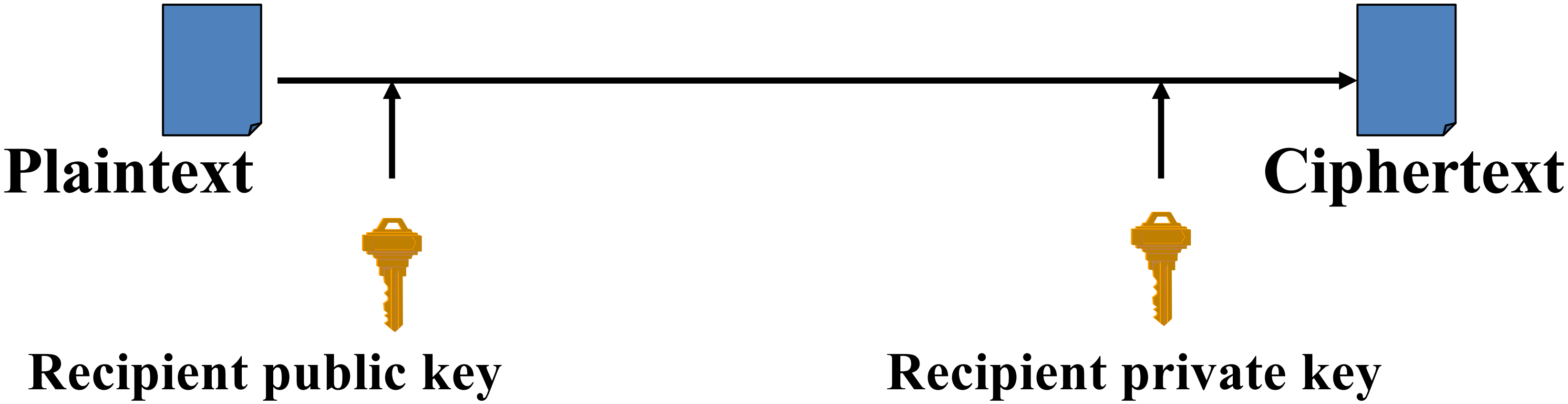
Notable usage: integrity protection, password « storage »

MD5 (do not use!), SHA-1 (DNU!), SHA-256, bcrypt, scrypt, Argon2, …

- 3 properties:
- Preimage resistance: given K, it is computationally hard to find M such that h(M)=K
- 2nd Preimage resistance: given M, it is computationally hard to find M' distinct from M such that h(M)=h(M')=K
- Collision-free: it is computationally hard to find M and M' distinct from M such that h(M)=h(M')
    - Birthday attacks …

- It is keyed if its computation depends on a secret information (termed MAC – Message Authentication Code)

# Asymmetric Key Cryptography




- Encryption based on the difficulty to find the inverse solution to a mathematical problem
  - Much more costly than symmetric key encryption

- Diffie-Hellman (1977)
  - Secret sharing : $(g^a)^b=(g^b)^a)=g^{ab}$ *mod p*
- RSA : Rivest-Shamir-Adleman (1978)
  - Standard solution today
  - $E_{KP}(P)=M^e$ *mod n* et $D_{KS}(C)=M^d$ *mod n*
  - *L*es clés KP=(e,n) et KS=(d,n) are connected by a mathematical relationship
- Elliptic curves: the new breed

# Diffie-Hellman algorithm

*p* is a large prime, *a* a primitive element of $Z_p^*$

| ***Known by A*** | ***Public*** | | ***Known by B*** |
|---|---|---|---|
| $0 < x \leq p-2$ | a,p | | $0 < y \leq p-2$ |
| $a^x$ mod p | $a^x$ mod p | $a^y$ mod p | $a^y$ mod p |
| $a^y$ mod p | | | $a^x$ mod p |
| compute : $(a^y)^x$ mod p | | | compute : $(a^x)^y$ mod p |
| $= a^{yx}$ mod p | | | $= a^{xy}$ mod p |

- ❒ A and B establish a shared secret ($a^{xy}$ mod p ) without exchanging any secret information
- ❒ $a^{xy}$ mod p may be used as a secret key with a symmetric key algorithm in order to encrypt data
- ❒ relies on the hardness to compute the discrete logarithm

Adapté d'après R. Molva

# Public key encryption

The algorithm on an operation whose inverse computation is hard

Modulus:

| | | | | | | |
|---|---|---|---|---|---|---|
| X | 1 | 2 | 3 | 4 | 5 | 6 |
| $3^x$ | 3 | 9 | 27 | 81 | 243 | 729 |
| $3^x$ mod 7 | 3 | 2 | 6 | 4 | 5 | 1 |

# RSA: Inversibility

**Euler Function**

For an integer n, $z = \phi(n)$ is the number of primes with n.

- if n is prime $\phi(n) = n-1$
- if $n = p.q$ with p and q prime

$$\phi(n) = (p-1)(q-1)$$

**Euler Theorem**

If a and n are respectivemy prime

$$a^{\phi(n)} \bmod n = 1$$

**Why RSA works**

$D_K(E_k\,(M)) = ((M)^e \bmod n)^d \bmod n$
$= (M^e)^d \bmod n = M^{e.d} \bmod n$

But we chose $e.d = 1 \bmod z$

Thus there exists an integer j such that $e.d = j\,z + 1$

$M^{e.d} = M^{j.z}\,M \bmod n = M \bmod n$

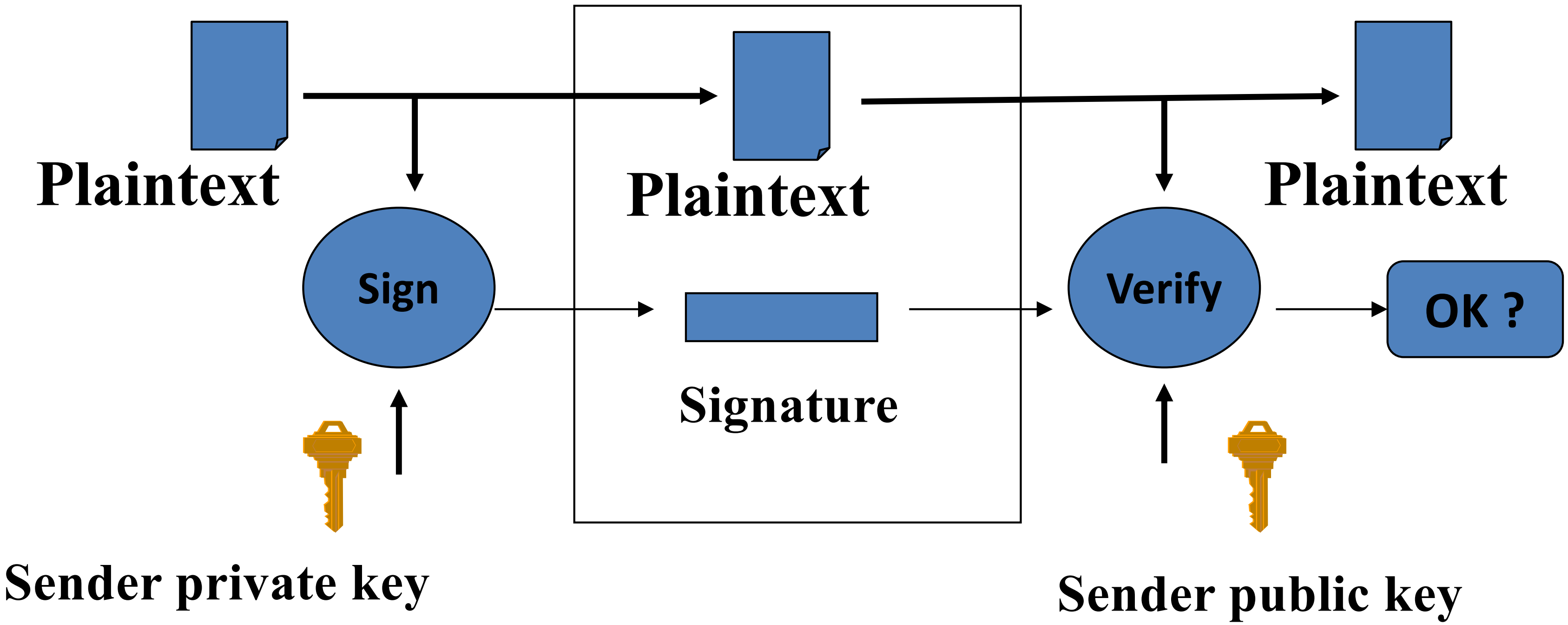
According to Euler theorem:

$M^{j.z} \bmod n = (M^z)^j \bmod n = (1)^j = 1$

# RSA: implementation weaknesses

- Never use a value overly small for *n*,
- Never use an overly short private key
- Use only strong keys such that p-1 and q-1 have a large prime factor
- Do not encrypt overly short blocks (always complete them to n-1 bits, so as to destroy any syntaxic structure
- Do not use a n factor common to several keys if those keys can be used to encrypt the same message
- If a private key *(d,n)* is compromised, do not use the other keys using *n* as a modulus
- Never encrypt or authenticate a message from a third party without modifying it (by adding a few random bytes for instance

# Digital Signature




- Comes with a message (which can be encrypted or in cleartext)
- Ensures:
  - The authentication of the origin of a message
  - The protection of the integrity of a message
  - The non repudiation of a message

# Signature using a public key algorithm

E, D : public key algorithm

Generating the signature of A over message M :

$$S = E_{KSa}(h(M))$$

Verifying the signature :

- compute h(M)
- verify whether $D_{KPa}(S) = h(M)$

# Cryptographic Libraries (and APIs)

- C:
  - OpenSSL: https://www.openssl.org
  - Cryptlib: https://www.cryptlib.com
  - NaCL (Networking and Cryptography): https://nacl.cr.yp.to
- Java:
  - JCA/JCE architecture and its implementations (Bouncy Castle: https://www.bouncycastle.org/fr/)
  - JAAS for authentication & authorization – notably for J2EE
  - Apache Commons Crypto (OpenSSL wrapper): https://github.com/apache/commons-crypto
  - jNaCL: https://github.com/neilalexander/jnacl
- Python:
  - NaCL (Networking and Cryptography): https://nacl.cr.yp.to
  - pyOpenSSL: https://www.pyopenssl.org/en/stable/index.html
  - cryptography: https://github.com/pyca/cryptography
- Web:
  - W3C low-level Crypto API (available in browsers): https://www.w3.org/TR/WebCryptoAPI/
  - IETF Javascript Object Signing and Encryption (jose): https://datatracker.ietf.org/wg/jose/about/

# Public Key Infrastructure (PKI)

- Public key certificates
  - Certifies the association of a name and a key
  - Some certificates can also reference authorizations (permissions)

- Foundational to today's secure and commercial Internet
- Assurances over identity, not trust!
  - Trust = external knowledge

- Example of usage: Thunderbird + Enigmail

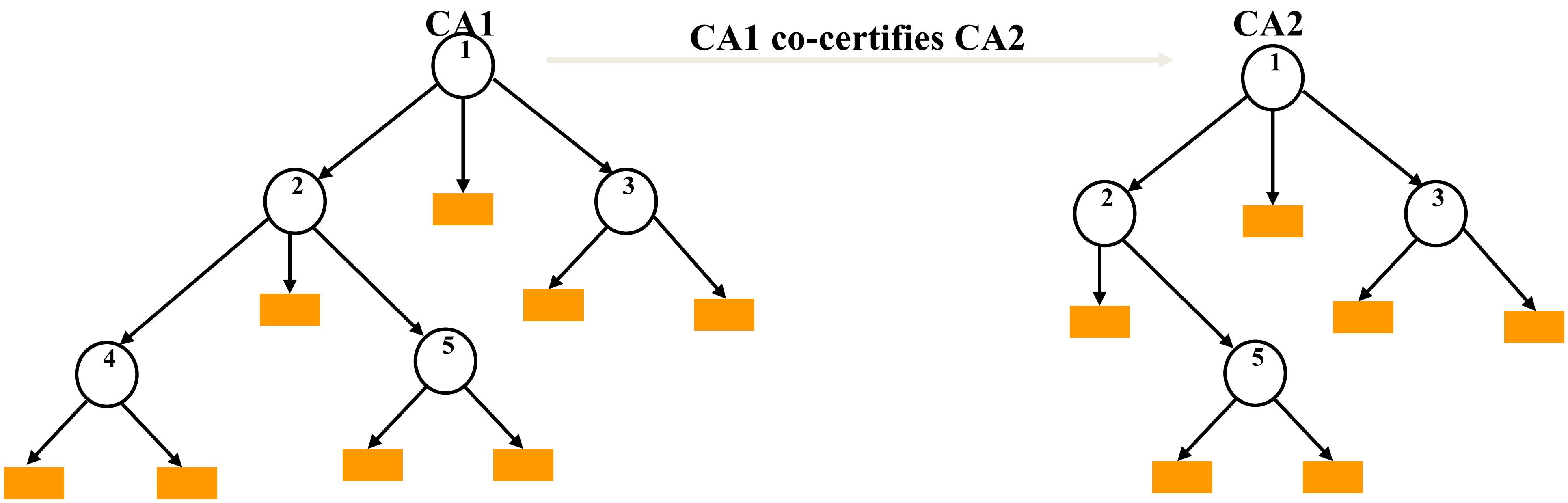
# Certificates

- Introduced in 1978 [Kohnfelder]
- Certificates ≠ Signature
  - Certificats are *implemented using* signatures
- Certificates ≠ Authentication / Authorizations
  - Authentication (resp. authorizations) *can be implemente with* certificates

- X.509 – the current "standard"
  - v.1 (1988) and v.2 – not extensible
  - v.3 (1997) current standard – optional extensions
  - Numerous other proposals extend X.509
- Other standards
  - PGP (notably OpenPGP), SPKI, etc.

# Trust Models in X.509

- Hierarchical Infrastructure
- Possibility to certify between 2 CAs belonging to different trees (co-certification)



# Countermeasures and More Vulnerabilities

- Countermeasures can lead to new vulnerabilities
  - E.g., if we only allow three incorrect logins, as a countermeasure to brute-force attacks (account is frozen), which new vulnerability do we introduce?
    - **Denial of Service attack**
  - If a countermeasure relies on new software, bugs in this new software may mean
    - that it is ineffective, or
    - worse still, that it introduces more weaknesses
    - E.g., Witty worm appeared in Mar 2004 exploited ISS security software
      - http://en.wikipedia.org/wiki/Witty_%28computer_worm%29

# Caveat: insecurities in cryptography

- SSH was meant to provide security, namely as countermeasure against eavesdropping on network, but is a source of new vulnerabilities

- Cryptography is not the cause of these vulnerabilities, and could not solve/prevent these vulnerabilities
  - Protocol, implementation errors (e.g., WEP in WiFi)
  - Programming errors (buffer overflow)
  - Distribution errors (trojan)

- Bruce Schneier: *“Currently encryption is the strongest link we have. Everything else is worse: software, networks, people. There's absolutely no value in taking the strongest link and making it even stronger”*