Cryptographie: théorie et pratique

- Mohamed Houcine HDHILI
- med_elhdhili@yahoo.es

Introduction à la cryptographie

…… …

Cryptographie: objectifs

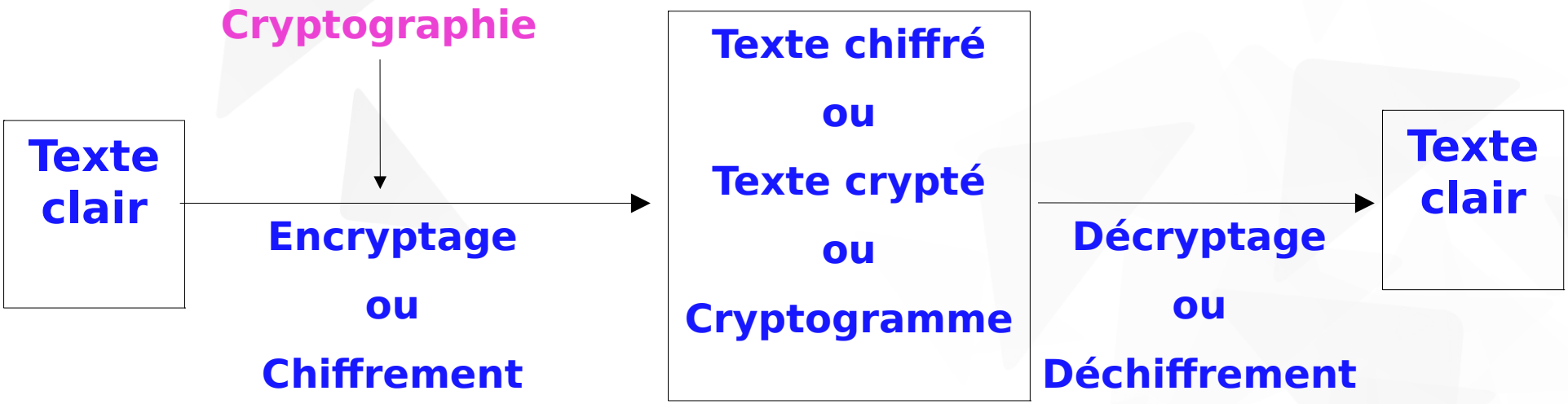
- Objectif principal:
  - Assurer la sécurité des communications sur un canal non sécurisé
-
- Sécurité des communications?
  - Confidentialité, Integrité, authenticité, non repudiation
-
- Canal non sécurisé?
  - Attaques passives: écouter des communications
  - Attaques actives: contrôler la communication (Man in the middle)
    - Injection,
    - Suppression,
    - modification

Terminologie

- Cryptologie= cryptographie+cryptanalyse
  -
  - Science (branche des mathématiques) des communications secrètes.
  -
  - Composée de deux domaines d'études complémentaires :
    -
    - Cryptographie : conception d'algorithmes/protocoles
    - Cryptanalyse : casser des algorithmes/protocoles

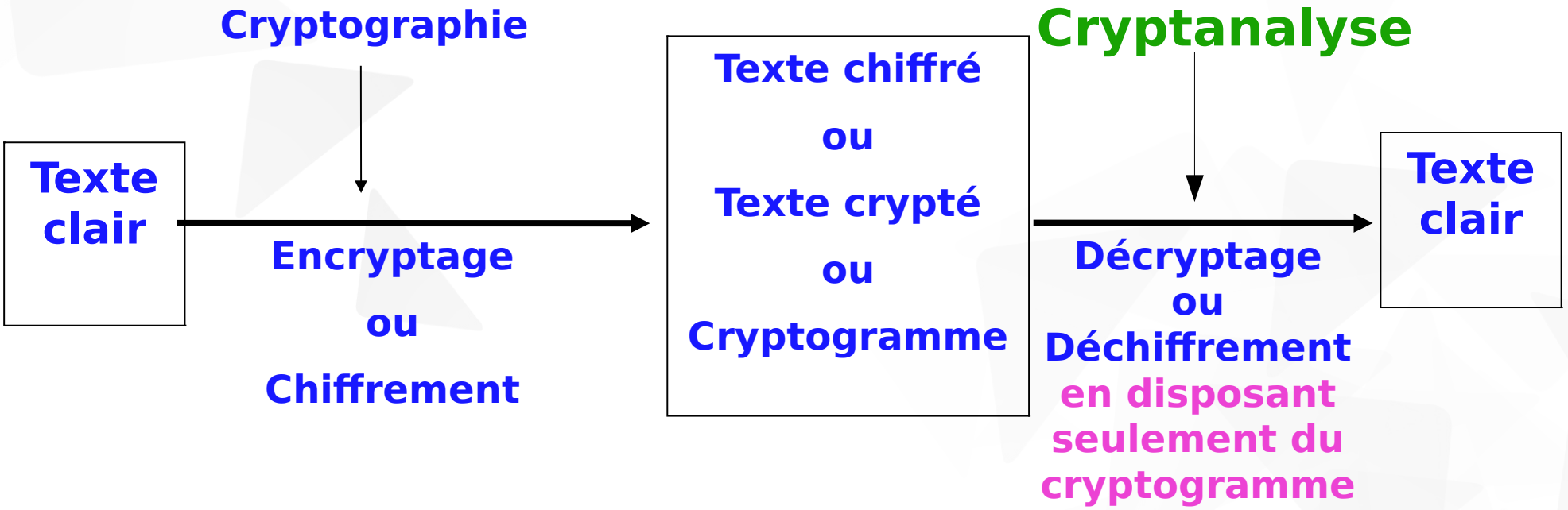
Terminologie

- Cryptographie (cryptography) = Chiffrement=Encryptage
  - Ensemble des méthodes et techniques qui permettent de transformer un message afin de le rendre incompréhensible pour quiconque qui n'est pas doté du moyen de le déchiffrer.
    - On parle d'***encrypter (chiffrer)*** un message,
    - Le code résultant s'appelle ***cryptogramme***.
    - L'action inverse s'appelle ***décryptage (déchiffrement)***.



Terminologie

- **Cryptanalyse (cryptanalysis)**
  - **Art de révéler les messages** qui ont fait l'objet d'un encryptage.
  - Lorsqu'on réussie, au moins une fois, à **déchiffrer un cryptogramme**, on dit que l'**algorithme** qui a servi à l'encrypter a été **cassé**.



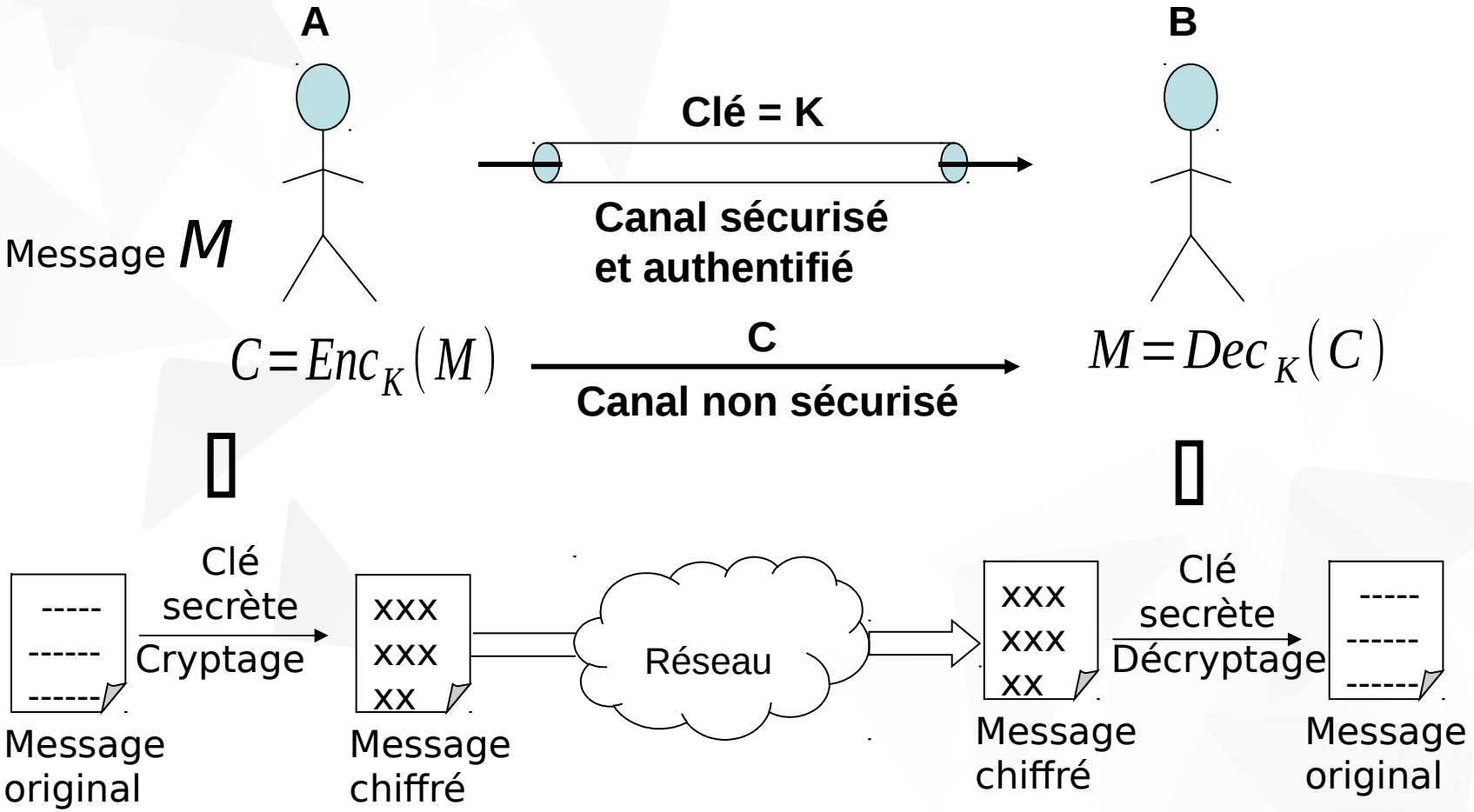
Terminologie

- Clé :
  - Information qui sera utilisée pour encrypter et / ou décrypter un message.

    On peut cependant concevoir un algorithme qui n'utilise pas de clé, dans ce cas c'est lui-même qui constitue le secret et son principe représente la clé
- Crypto système:
  - Ensemble composé
    - d'un algorithme,
    - de tous les textes en clair (**espace des textes clairs**),
    - de tous textes chiffrés (**espace des textes chiffrés**)
    - de toutes les clés possibles (**espace des clés**).

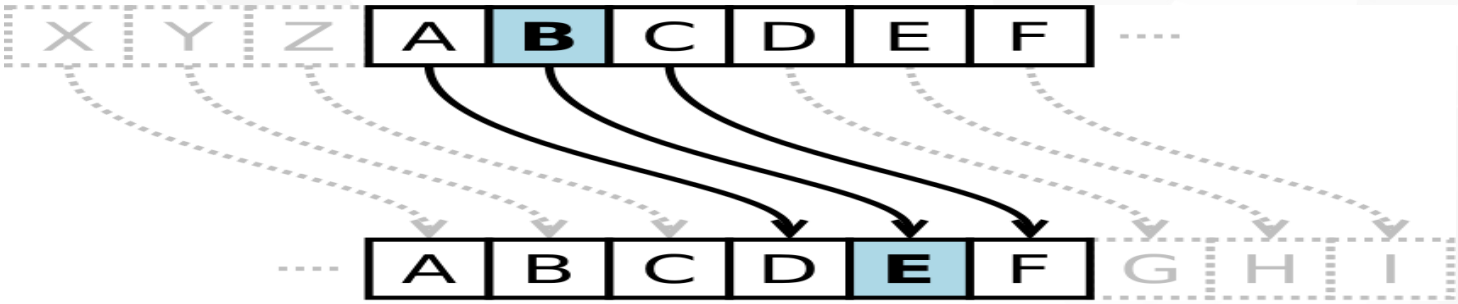
Chiffrement

- Chiffrement symétrique



Chiffrement par décalage (shift cipher)

- Exemple: espace des clés [0..25]
- Chiffrement en utilisant **K**
  - Chaque lettre du message clair est remplacé par la k^{ieme} lettre en partant de la lettre à remplacer (shift right)
- Déchiffrement:
  - shift left
- Cryptanalyse:
  - **Il est facile de déterminer K (brute force attack: tester les 26 possibilités)**
- Exemple: chiffrement de cesar (K=3)

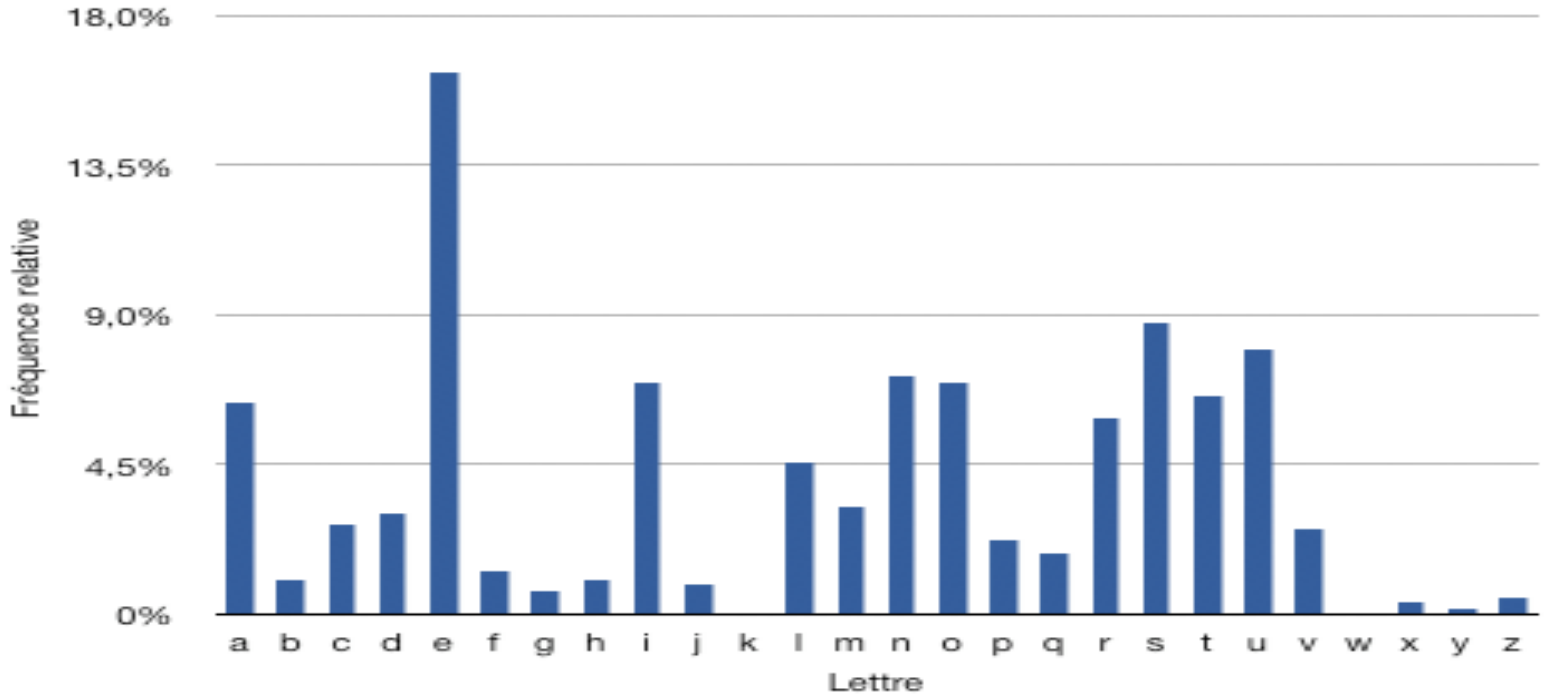


Chiffrement mono alphabétique par substitution

- Espace des clés:
  - Exemple: toutes les permutations de Σ={A,B,.....Z}
- Chiffrement en utilisant une clé K
  - Chaque lettre X du message clair est remplacé par la lettre K(X)
- Déchiffrement en utilisant la clé K
  - Chaque lettre Y du message chiffré est remplacé par $K^{-1}(Y)$
- Exemple:
  - ESSALAMO ALEIKOM ==> WNNSYSPF SYEVQFP

Chiffrement mono alphabétique par substitution: Cryptanalyse

- La recherche exhaustive est difficille: Espace des clés= 26!
- Cryptanalyse par analyse fréquentielle: facile
  - Découverte par les arabes (Al KINDI):
    - Idée: Chaque langage possède certaines caractéristiques: fréquences des lettres, de groupes de 2 lettres....
  - ==> insécurité du chiffrement par substitution
  - Exemple: fréquences des lettres en Français



Défense contre l'analyse fréquentielle

- Utiliser des blocs plus long (64 ou 128 bits) pour la substitution au lieu de faire la substitution lettre par lettre
  - ==> exemple de chiffrement par bloc: DES, AES...
- Utiliser différents substitution
  - Faire correspondre à chaque caractère du texte en clair un ensemble d'équivalents possibles appelé homophones (généralement des chiffres)
  - ==> analyse fréquentielle difficille mais possible
- Chiffrement par substitution polyalphabétique
  - ...

Evolution de la cryptographie classique

...

Chiffrement par substitution polyalphabétique

- Faiblesse du chiffrement monoalhabétique
  - Chaque lettre du message chiffré correpond à une seule lettre du message clair
  - ==> possibilité de l'analyse fréquentielle
-
- Solution: substitution polyalphabétique
  - Utiliser plusieurs alphabets de chiffrement et les utiliser lors du chiffrement de différentes lettres
    - ==> **les fréquences des lettres du cryptogramme sont similaires**
  - Exemple: chiffrement de viginère (1586)

Chiffrement de viginère

- Traiter les lettres comme nombres: {A=0, B=1,...Z=25}
- Notation Z_n={0, 1, 2,...(n-1)}
- Définition:
  - Soit :
    - P: plaintext, C: Ciphertext
    - m un entier >0, P=C=$(Z_{26})^n$ et K={k_1, k_2, ..k_m}
  - Encryption: $e_k(p_1, p_2... p_m) = (p_1+k_1, p_2+k_2...p_m+k_m)$ (mod 26)
  - Decryption: $d_k(c_1, c_2... c_m) = (c_1- k_1, c_2- k_2...c_m - k_m)$ (mod 26)
  - Exemple
    - Plaintext: ATTACKATDAWN
    - Key: **LEMON**LEMONLE (la clé est re-écrite)
    - Ciphertext: LXFOPVEF**R**NH**R**

Tableau de viginère

Lettres du message

Clés ki

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
B	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
C	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
D	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
E	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
F	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
G	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
H	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
I	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
J	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
K	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j
L	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
M	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l
N	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
O	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
P	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
Q	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
R	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
S	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
T	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
U	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
V	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
W	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
X	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
Y	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
Z	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Fig.2 Tableau de Vigenère

Chiffrement de viginère: sécurité et cryptanalyse

- Sécurité:
  - La fréquence d'apparition des lettres est masqué ==> complique l'analyse fréquentielle
  - Un chifrement viginère = collection de plusieurs chiffrement par décalage (plusieurs ie la longueur de la clé)
-
- Cryptanalyse: Quoi faire?
  - Trouver la longueur de la clé
    - Kasisky test
    - Index of coincidence (Freidman)
  - Diviser le message en plusieurs substitutions simples à résoudre
  - Comment?

Kasisky test (1863)

- Constat:
  - Deux segments identiques du texte clair donnent le même cryptogramme s'ils apparaissent dans le texte à la distance Δ (Δ=0 mod m). m est la longueur du mot clé.
  - Algorithm
    - Rechercher les paires de segments identiques de longueur au minimum 3
    - Enregistrer les distances Δ1, Δ2,...
    - M divise le PGCD(Δ1, Δ2,...).

Test de KASISKI: exemple

- Wakeupalicedearsaidh
  ersisterwhywhatalong
  sleepyouvehadohiveha
  dsuchacuriousdREAmsAI
  DaliceandshetolDHER
  SISTERaswellasshecou
  Ldrememberthemallthe
  Sestrangeadventureso
  Fhersthatyouhavejust
  BeenREAdingaboutandw
  HenSHEhadfiniSHEDHER
  SISTERkissEDHerandsA
  Iditwasacuriousdream
  dearcertainlybutnowr
  unintoyourteaitsgettinglate

- Hegmmaehquphaijdeelz
  pvoqkeinezjadillpkvy
  dpamhjsqdwsezwztzaps
  owqkzlgqzazyol**JPE**ia**S**
  THwtaniwvvdlabgw**HDMJ**
  DMOBWCeoewwpwaksiywm
  whnmepqxmjelauswpppw
  diobjlrcmsozavlfvaag
  qlazkelwbqzydinpnqal
  miav**JPE**zqfrexwmeejlo
  sij**AZP**lwlxtre**AZP**HDMJ
  DMOBWCoeakPHDmjlrza**S**
  THebolwwkmcmkckovaie
  oiwzupvpiaypujmerkej
  frevlzckcjeiwqldkabltrctsei

- Distances observées:
  - JPE(110), STH(160), HDMJDMOBWC(120), PHD(15), AZP(10)
  - ==> longueur de la clé a une forte probabilité d'être PGCD(110,160,120,15,10)=5

One Time PAD and perfect secrecy

…… …

One Time PAD (Verman 1920)

- Résoud la faiblesse du chiffrement de viginère
  - Solution: taille de la clé >= taille du message
  - Une clé est utilisé pour chiffrer un seul message (One Time)
- Soit l'alphabet Z_m={0, 1, 2,...(m-1)}
  - Espace des clés=espace des textes clair=espace des textes chiffrés= $(Z_m)^n$
  - **P={p_1, p_2, ..p_n}, K={k_1, k_2, ..k_n} et C={c_1, c_2, ..c_n}**
    - Encryption: **$e_k(p_1, p_2 ... p_n) = (p_1+k_1, p_2+k_2 ... p_n+k_n)$ (mod m)**
    - Decryption: **$d_k(c_1, c_2 ... c_n) = (c_1- k_1, c_2- k_2 ... c_n - k_n)$ (mod m)**
- One Time PAD has perfect secrecy (proved)
  - Le texte chiffré ne donne aucune idée sur le texte clair

Perfect secrecy

- Le texte chiffré ne doit donner aucune idée sur le texte en clair

  ➔ Les deux variables aléatoires P (plaintext) and C (ciphertext) sont indépendantes
- Définition: (GEN, ENC, DEC) sur un espace S de messages est parfaitement secret si

  ∀ distribution de probabilité sur S

  ∀ message msg sur S

  ∀ texte chiffré c∈ C avec Pr[C=c] > 0, on a

  Pr [M=msg | C=c] = Pr [M = msg].

Ou : Pr [M=msg et C=c] = Pr [M = msg].Pr [C = c]

Ou : Pr [C=c | M=msg] = Pr [C = c].

Ou : Pr [C=c | M=msg1] = Pr [C=c | M=msg2] ∀ msg1, msg2 sur S

Usage de One Time PAD

- Taille de la clé >= taille du message
- La clé est utilisé une seule fois (One Time)

➔ il est plus adéquat d'échanger le message sur un canal sécurisé que d'échanger la clé pour chiffrer le message

- OTP reste utile
  - La distribution des clés peut se faire au préalable (avant d'avoir des messages à envoyer)

Stream ciphers

…… …

Stream ciphers

- Dans One Time PAD:
  - Taille de la clé >= taille du message
  - Clé= chaine aléatoire
-
- Stream ciphers
  - Idée: remplacer “aléatoire” par “pseudo aléatoire”
  - Utiliser un **PRNG (Pseudo Random Number Generator)**
    - G: $\{0,1\}^s$--> $\{0,1\}^n$ avec n>>s
    - Étendre une clé aléatoire K de petite taille (exemple 128bits) pour avoir une clé pseudo aléatoire de grande taille (exemple 10^6 bits)
  - **Chiffrement: $E_K(M)=M \oplus G(K)$**
- **Exemple: RC4**

Message Authentication Code (MAC)

…… …

Intégrité des données et authentification de la source

- Integrité:
  - Les données doivent arriver au destinataire dans leur forme originale envoyé par la source
- Authentification de la source
  - Les données proviennent d'une source authentique
- MAC
  - La source et le destinataire partage une clé K qui est utilié pour l'authentification des messages
    - Key generation: **$k=GEN(1^n)$**
    - Tag generation: **$t=MAC_k(m)$**
    - Verification algorithm: **$b=verify_k(m,t)$** (b=1 sig verification valide)

CBC MAC (fixed length MAC)

- Soit une fonction pseudo aléatoire PRF, et L(n) une fonction longueur
- PRF $F:\{0,1\}^n x \{0,1\}^n \rightarrow \{0,1\}^*$
- CBC MAC est:
  - $mac_K(m)$, m est de longueur L(n).n
    - Diviser m en $m_1, m_2, \ldots m_L$ chacune de taille n
    - $t_0=0^n$
    - **Pour i de 1 à L, $t_i=F_K(t_{i-1} \oplus m_i)$**
    - **Output: t_L**
  - Verify (K, m, t)? ==> vérifier si **$mac_K(m)=t_L$**
- Proved to be secure

Hash functions

- Une fonction de hashage génère à partir d'un message de **longueur quelconque** un résumé de **taille fixe**
-
- fonction de hashage cryptographique **h: X➔Y** est:
  - **preimage resistant (one way):**

    étant donné $y \in Y$, il est impossible de trouver $x \in X$ tel que $h(x)=y$
  - **2nd preimage resistant (weak collision resistant):**

    étant donné $x \in X$, il est impossible de trouver $x' \in X$ tel que $x' \neq x$ et $h(x')=h(x)$
  - **collision resistant (strong collision resistant):**

    Il est impossible de trouver deux valeurs distinctes x et x' tel que $h(x)=h(x')$

Brute force attacks on Hash functions

- Attacking one-wayness
  - Objectif:

    étant donné **h: X➔Y** et **y∈ Y**, trouver **x∈ X** tel que **h(x)=y**
  - Méthode :

    Prendre une valeur aléatoire **x∈ X** et voir si **h(x)=y ?** si oui afficher x sinon itérer (au max q fois)
  -
- Attacking collision resistance
  - Objetcif:

    trouver **x∈ X et x'∈ X**, tel que **x'≠x et h(x')=h(x)**
  - Méthode:

    Tester **q** valeurs disctinctes de **X** et voir si l'objetcif est atteint
  -

Fonction de hashage importantes

- MD5 (128 bits)
  - Resistance aux collisions
    - cassé (Prof. Xiaoyun Wang, china 2004)
- SHA1 (160 bits)
  - “One way” est encore valide
  - Resistance aux collisions
    - Considéré non sécurisé (pas de collisions déterminées mais des méthodes existent en 2^80)
- SHA2 (SHA224, SHA-256, sha-384, SHA-512)
  - Donnent 224, 256, 384, 512 bits respectivement
  - Sécurisée
- SHA3(SHA3-224, SHA3-256, SHA3-384, SHA3-512)
  - sécurisée

MAC et fonctions de hashage: Applications

- Challenge-response
  - Objectif: une entité s'authentifie à une autre prouvant la possession d'un secret (challenge).
  - Approche: utiliser des paramètres qui dépendent du temps (nonce, timestamp) ==> prévenir les attaques de rejeu
  - Exemple: challenge-response basé sur un chiffrement symétrique
    - Authentification unilatérale avec timestamp: **A vers B: $MAC_K(T_A, B)$**
    - Authentification unilatérale avec nonce:
      - **B vers A: R_B**
      - **A vers B: $MAC_K(R_B, B)$**
    - Authentification mutuelle avec nonce
      - **B vers A: R_B**
      - **A vers B: R_A , $MAC_K(R_A, R_B, B)$**
      - **B vers A: $MAC_K(R_B, R_A)$**

MAC et fonctions de hashage: Applications

- Utiliser des mdp sur des canaux non sécurisés
- Integrité d'un logiciel
- Stocker le hash du mdp au lieu du mdp lui même
  - Exemple: /etc/shadow
- One Time password
  - La source et le destinataire sauvegarde plusieurs mdp
  - Chaque mdp est utilisé une seule fois
  - Exemple: utiliser une chaine de hashage (Lamport) dans l'ordre inverse

    $h(s), h(h(s)), h(h(h(s)))..... h^{1000}(s)$

Chiffrement par Blocs

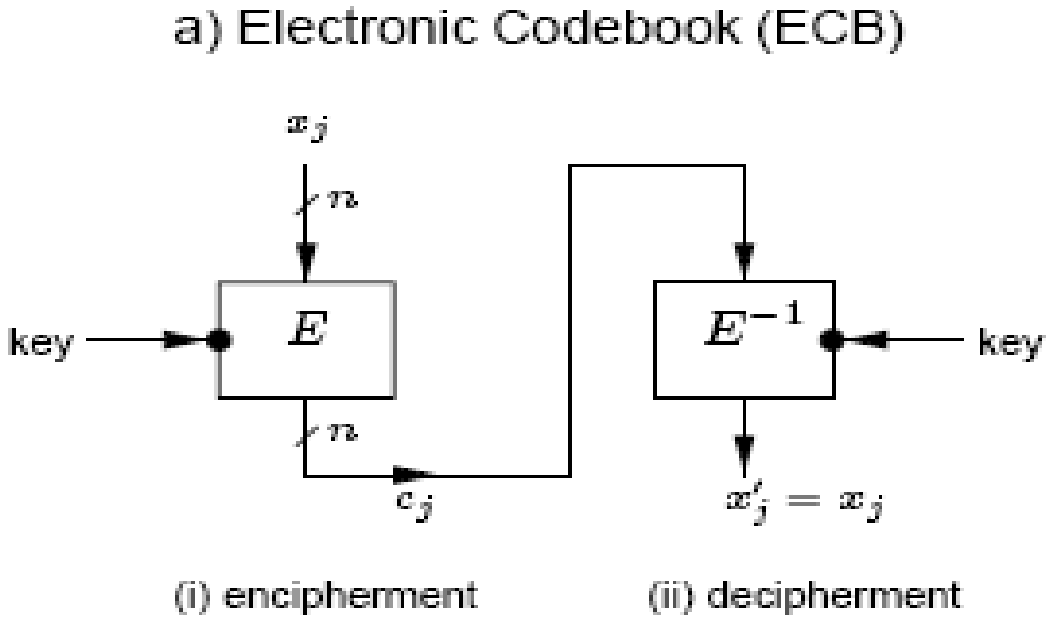
…… …

Chiffrement par blocs

- Méthode:
  - Le message M à chiffrer est scindé en un nombre de bloc de taille fixe: **$M=x_1, x_2, ..x_n$**
  - Cryptage des blocs
  - Le cryptogramme **C** est obtenu en concaténant les cryptogrammes des blocs
-
- Modes de chiffrement par bloc
  - ECB (Electronic CodeBook)
  - CBC(Cipher bloc Chaining)
  - CBF (Cipher FeedBack)
  - OFB (Output FeedBack)

ECB (Electronic CodeBook)

- Chaque bloc est chiffré à part



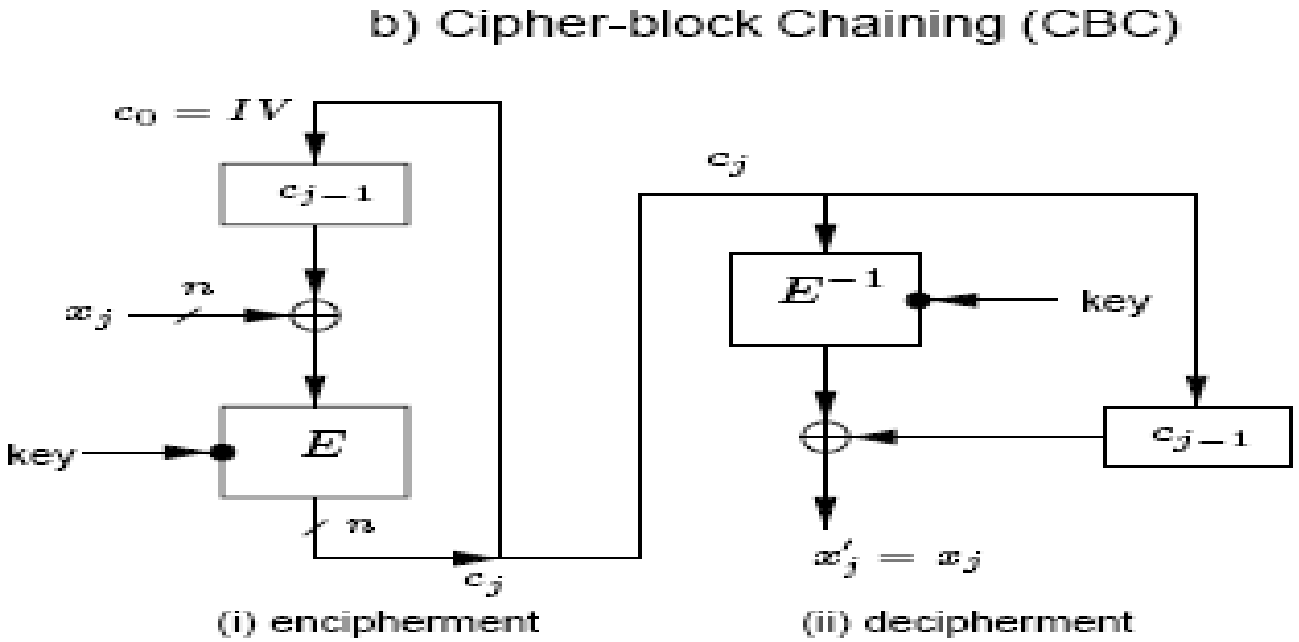
7.11 Algorithm ECB mode of operation

INPUT: k-bit key K; n-bit plaintext blocks $x_1, \ldots, x_t$.
SUMMARY: produce ciphertext blocks $c_1, \ldots, c_t$; decrypt to recover plaintext.

1. Encryption: for $1 \leq j \leq t$, $c_j \leftarrow E_K(x_j)$.
2. Decryption: for $1 \leq j \leq t$, $x_j \leftarrow E_K^{-1}(c_j)$.

CBC(Cipher bloc Chaining)

- Chaque bloc du cryptogramme dépend du bloc de texte en clair et de tous les blocs précédents



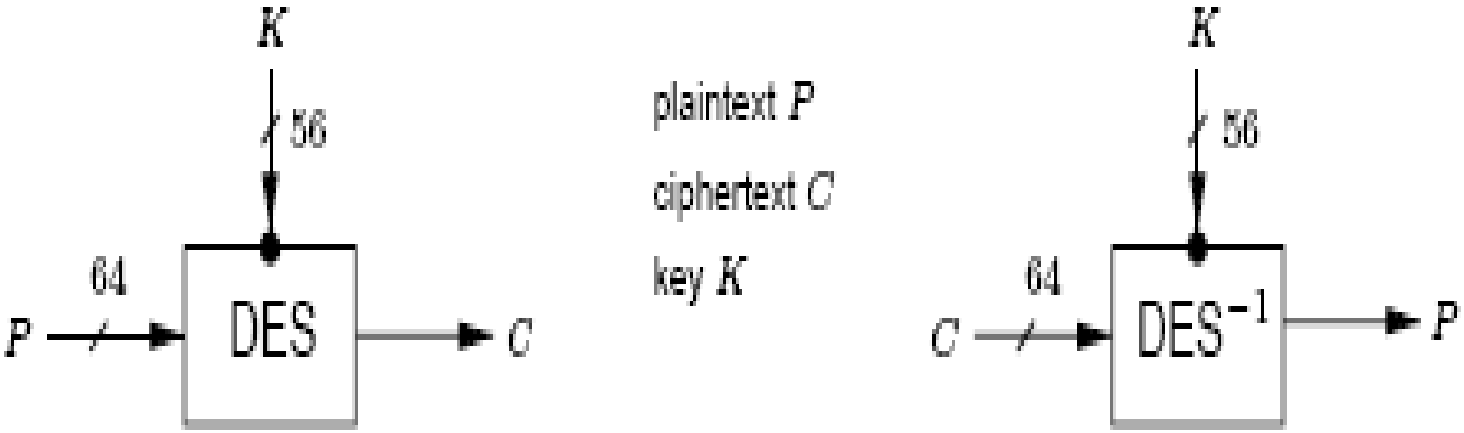
7.13 Algorithm CBC mode of operation

INPUT: k-bit key K; n-bit IV; n-bit plaintext blocks $x_1, \ldots, x_t$.
SUMMARY: produce ciphertext blocks $c_1, \ldots, c_t$; decrypt to recover plaintext.

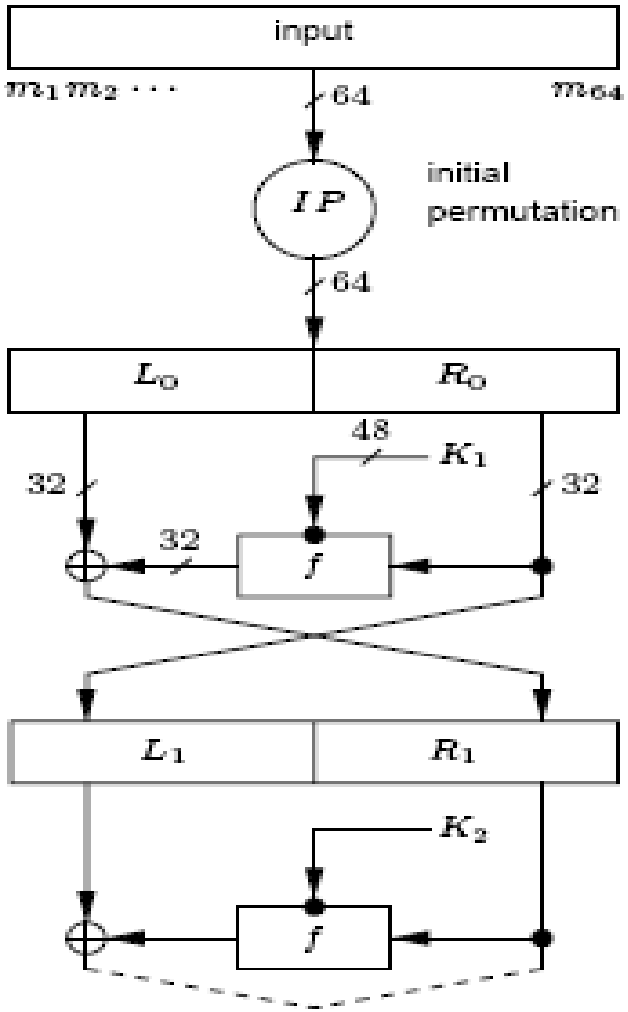
1. Encryption: $c_0 \leftarrow IV$. For $1 \leq j \leq t$, $c_j \leftarrow E_K(c_{j-1} \oplus x_j)$.
2. Decryption: $c_0 \leftarrow IV$. For $1 \leq j \leq t$, $x_j \leftarrow c_{j-1} \oplus E_K^{-1}(c_j)$.

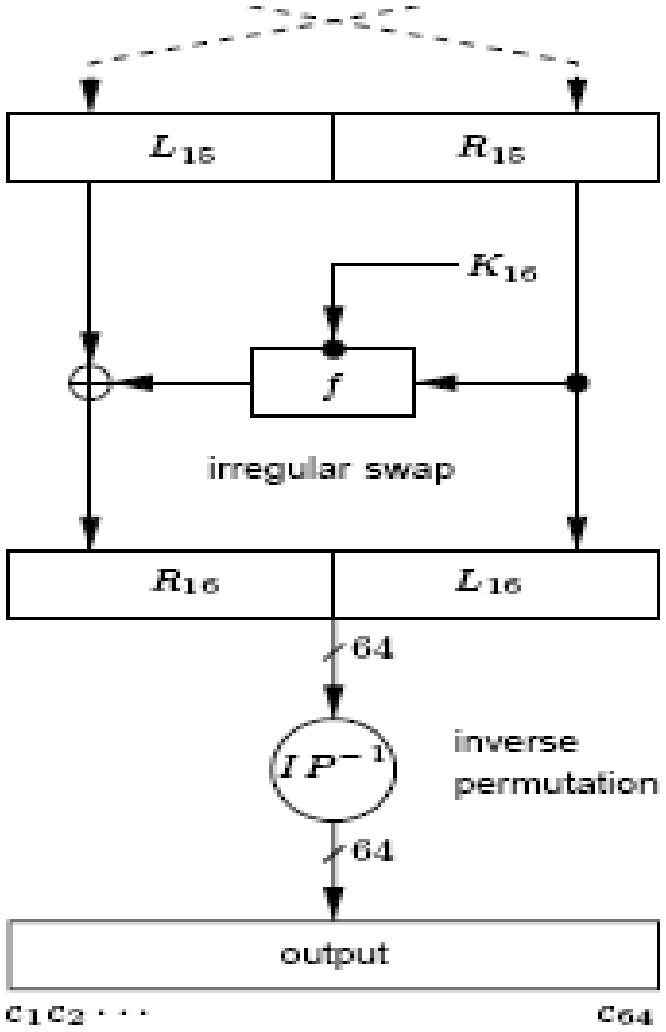
DES (Data Encryption Standard)

- Chiffrement **par bloc** : 64bits
- Clé de taille variable:
  - Entre 56 et 128 bits selon le niveau de sécurité désiré
  - Version initiale du DES utilise une clé de taille 64 bits dont 56 sont réellement utilisés
- Utilise un **chiffrement produit**
  - Combine des algorithmes de **substitution** et des algorithmes de **transposition** ➔ maximiser la complexité de l'algorithme



DES (Data Encryption Standard)





$$L_i = R_{i-1};$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i), \quad \text{where } f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$$

Particularités du DES

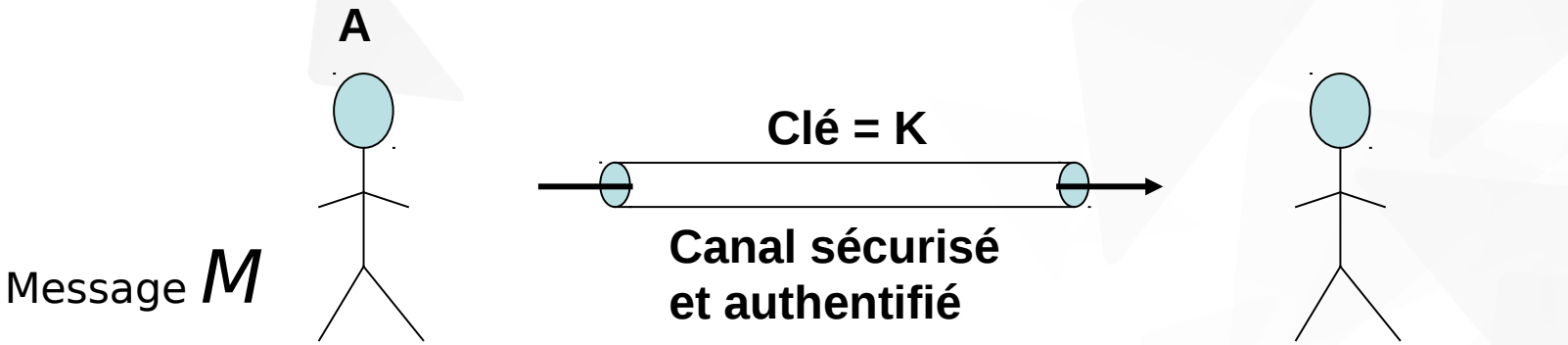
- Souplesse d'implémentation:
  - ECB ou CBC (en modifiant la phase de pré traitement des blocs de données)
  - Différentes implémentation en modifiants les fonctions d'expansion ou de sélection
- Faiblesses
  - Conservation de la taille ➔ sensible aux attaques d'analyse de flux: on peut connaitre la taille exacte de chaque message
  - La clé est réduite à 56 bits ➔ réduit la sécurité de l'algorithme
  - Avec une clé de taille 128 bits ➔ algorithme couteux en temps
  - **Peut être cassé par les processeurs actuels (exhaustive key search)**
  - Triple DES (trois clés différentes)
  - AES : remplaçant du DES

Gestion de clés...
Vers la cryptographie à clé publique

…… …

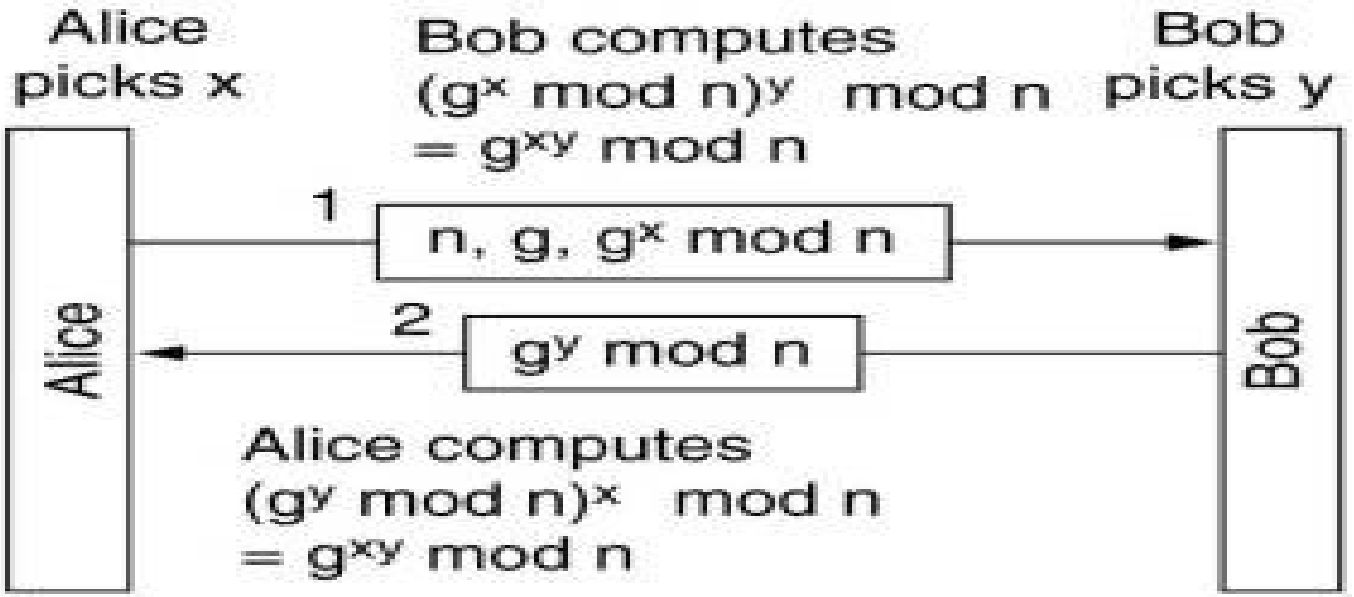
chiffrement symétrique (revisited)

- Comment établir la clé K d'une manière sécurisé **?**
  - Key transport : l'un crée la clé et la transmet à l'autre
  - Key agreement : une clé partagée est dérivée par 2 entités (ou +)
- Comment mettre à jour la clé K ?
  - Clé de session : temporaire
  - Clé à long terme
- Besoin de **N(N-1)/2 clés** pour N utilisateur => trop de clés
- ==> Sol : utiliser un centre de distribution de clé (KDC)
  - Chaque entité partage une clé avec le centre



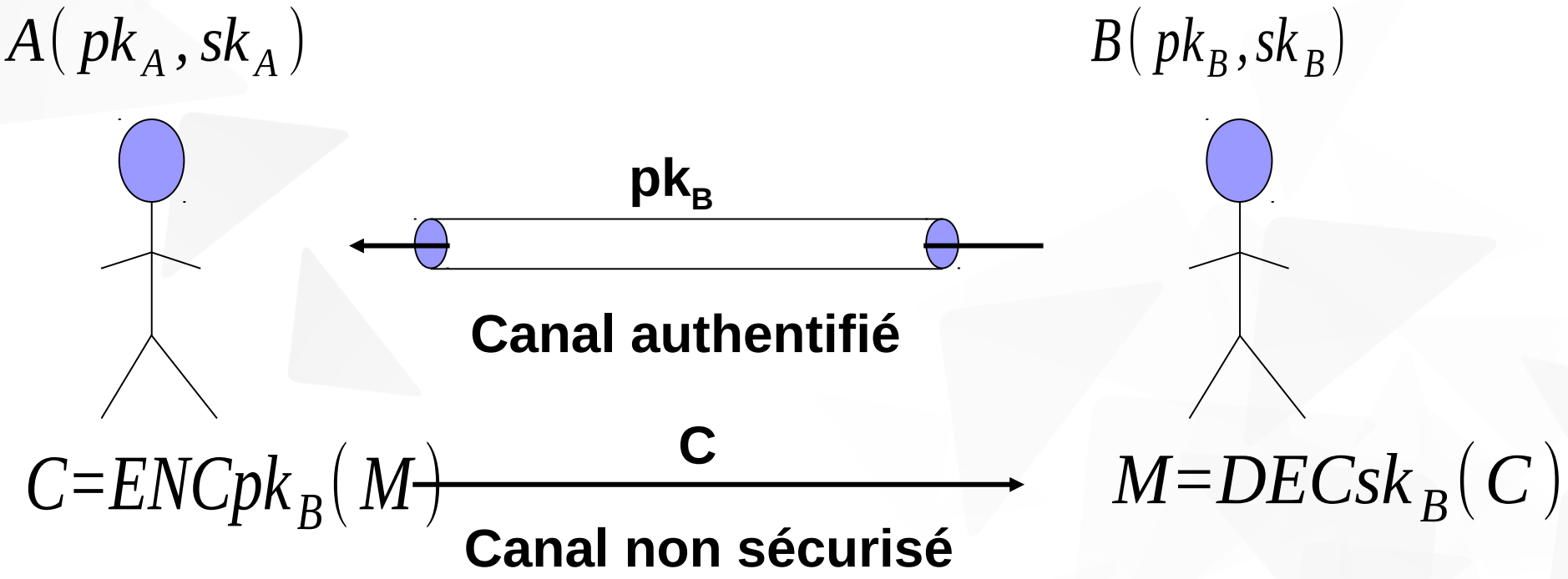
Diffie-Hellman key agreement protocol

- Etablir une clé commune via un canal publique
  - p : nombre premier, g : générateur de Zp* ==> publiques
- Basé sur deux problèmes difficiles
  - Discrete Log Problem(DLP) : étant donné **(g, h, p)** calculer **a** tel que $\mathbf{g^a = h \bmod p}$
  - Computational Diffie Hellman (CDH):étant donné $\mathbf{g^a \bmod p}$ et $\mathbf{g^b \bmod p}$ calculer $\mathbf{g^{ab} \bmod p}$



Chiffrement asymétrique (à clé publique)

- Chaque entité posède deux clés
  - **Pk**: public key ==> clé disponible pou tout le mo de
  - **Sk**: secret (private) key ==> clé privée de l'entité
  - Il est imposible de calculer **Sk** en connaissant **Pk**



Cryptosystèmes asymétriques

public-key encryption scheme	computational problem
RSA	integer factorization problem (§3.2) RSA problem (§3.3)
Rabin	integer factorization problem (§3.2) square roots modulo composite n (§3.5.2)
ElGamal	discrete logarithm problem (§3.6) Diffie-Hellman problem (§3.7)
generalized ElGamal	generalized discrete logarithm problem (§3.6) generalized Diffie-Hellman problem (§3.7)
McEliece	linear code decoding problem
Merkle-Hellman knapsack	subset sum problem (§3.10)
Chor-Rivest knapsack	subset sum problem (§3.10)
Goldwasser-Micali probabilistic	quadratic residuosity problem (§3.4)
Blum-Goldwasser probabilistic	integer factorization problem (§3.2) Rabin problem (§3.9.3)

Cryptosystème RSA (Ron Rivest, Adi Shamir and Leonard Adleman 1978): génération des clés

Algorithm Key generation for RSA public-key encryption

SUMMARY: each entity creates an RSA public key and a corresponding private key. Each entity A should do the following:

1. Generate two large random (and distinct) primes p and q, each roughly the same size.
2. Compute $n = pq$ and $\phi = (p-1)(q-1)$. (See Note 8.5.)
3. Select a random integer e, $1 < e < \phi$, such that $\gcd(e, \phi) = 1$.
4. Use the extended Euclidean algorithm (Algorithm 2.107) to compute the unique integer d, $1 < d < \phi$, such that $ed \equiv 1 \pmod{\phi}$.
5. A's public key is (n, e); A's private key is d.

2.107 Algorithm Extended Euclidean algorithm

INPUT: two non-negative integers a and b with $a \geq b$.
OUTPUT: $d = \gcd(a, b)$ and integers x, y satisfying $ax + by = d$.

1. If $b = 0$ then set $d \leftarrow a$, $x \leftarrow 1$, $y \leftarrow 0$, and return(d,x,y).
2. Set $x_2 \leftarrow 1$, $x_1 \leftarrow 0$, $y_2 \leftarrow 0$, $y_1 \leftarrow 1$.
3. While $b > 0$ do the following:
   3.1 $q \leftarrow \lfloor a/b \rfloor$, $r \leftarrow a - qb$, $x \leftarrow x_2 - qx_1$, $y \leftarrow y_2 - qy_1$.
   3.2 $a \leftarrow b$, $b \leftarrow r$, $x_2 \leftarrow x_1$, $x_1 \leftarrow x$, $y_2 \leftarrow y_1$, and $y_1 \leftarrow y$.
4. Set $d \leftarrow a$, $x \leftarrow x_2$, $y \leftarrow y_2$, and return(d,x,y).

Cryptosystème RSA: chiffrement

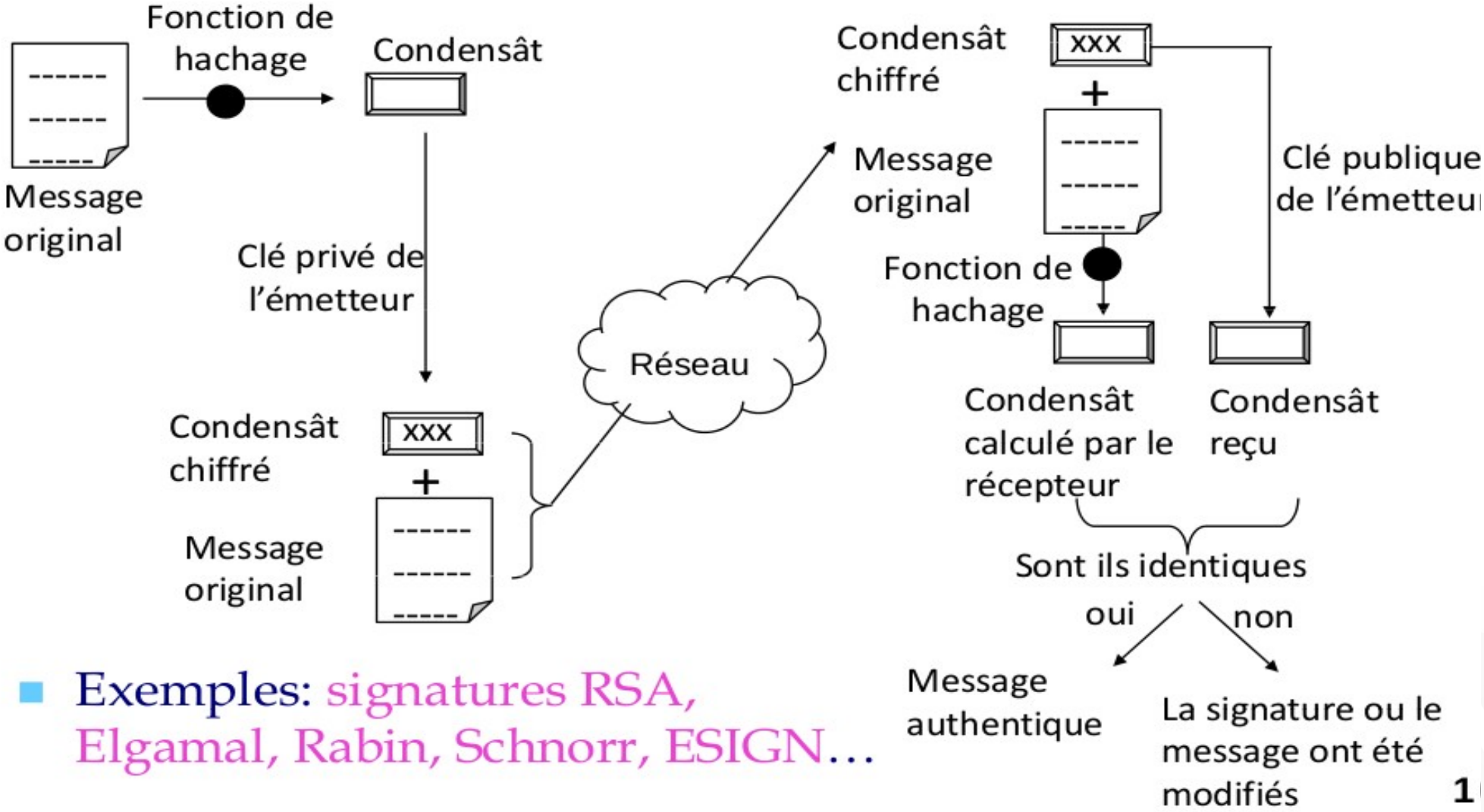
Algorithm RSA public-key encryption

SUMMARY: B encrypts a message m for A, which A decrypts.

1. *Encryption.* B should do the following:
   (a) Obtain A's authentic public key (n, e).
   (b) Represent the message as an integer m in the interval $[0, n-1]$.
   (c) Compute $c = m^e \bmod n$ (e.g., using Algorithm 2.143).
   (d) Send the ciphertext c to A.
2. *Decryption.* To recover plaintext m from c, A should do the following:
   (a) Use the private key d to recover $m = c^d \bmod n$.

Signatures digitales

- Permet l'authentification, l'intégrité et la non répudiation



- Exemples: signatures RSA, Elgamal, Rabin, Schnorr, ESIGN…

N.B: on peut signer un message sans appliquer une fonction de hashage

Signature RSA

11.19 Algorithm RSA signature generation and verification

SUMMARY: entity A signs a message $m \in \mathcal{M}$. Any entity B can verify A's signature and recover the message m from the signature.

1. *Signature generation.* Entity A should do the following:
   (a) Compute $\bar{m} = R(m)$, an integer in the range $[0, n-1]$.
   (b) Compute $s = \bar{m}^d \bmod n$.
   (c) A's signature for m is s.
2. *Verification.* To verify A's signature s and recover the message m, B should:
   (a) Obtain A's authentic public key (n, e).
   (b) Compute $\bar{m} = s^e \bmod n$.
   (c) Verify that $\bar{m} \in \mathcal{M}_R$; if not, reject the signature.
   (d) Recover $m = R^{-1}(\bar{m})$.

“TextBook” RSA Signature

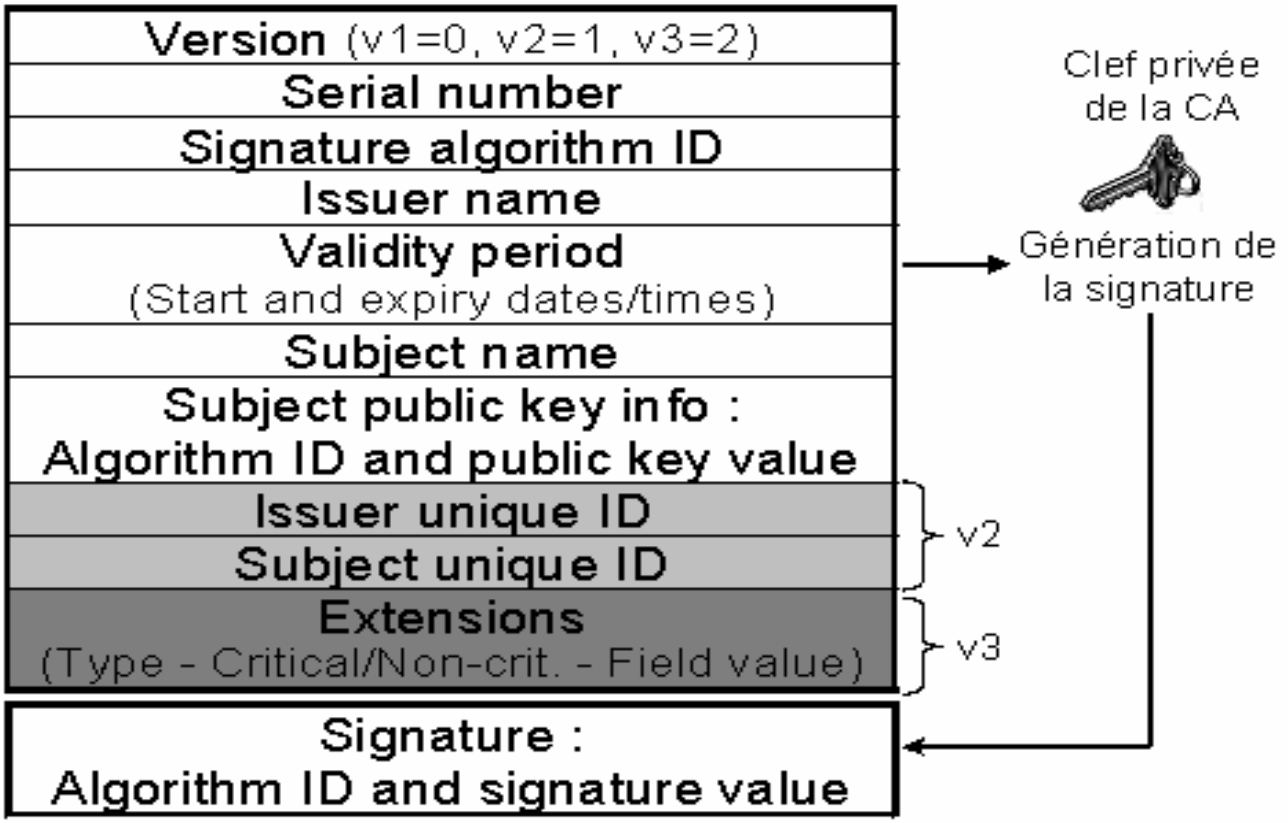
- message m, clé privé {d, n}, clé publique {e,n}
- Signature **$S = m^d \bmod n$**
- Vérification de la signature: vérifier que **$S^e \bmod n = m$**
-
- **Non sécurisé:**
  -
  - Attaque 1: “ no message attack”
    - Choisir une valeur arbitraire de S et calculer **$S^e \bmod n = m$**
    - **(m, S) est valide ==> on ne peut pas contrôler m ????**
  -
  - Attaque 2: “forge signature on arbitrary message”
    - Demander la sigature de **m1** et **m/m1 mod n** pour obtenir **S1** et **S2**
    - (m, S1.S2) est valide

Signature avec hashage

- message m, clé privé {d, n}, clé publique {e,n}
  - Utiliser une fonction de hashage $H:\{0,1\}^* \rightarrow Z_n^*$
    - Signature **$S = [H(m)]^d \bmod n$**
    - Vérification de la signature: vérifier que **$S^e \bmod n = H(m)$**
- Sécurisée: pas d'attaques existants

Certificat

- M Permet l'authentification
  - Garantit l'appartenance d'une clé publique à une entité
- Principal format: **certificats X.509**

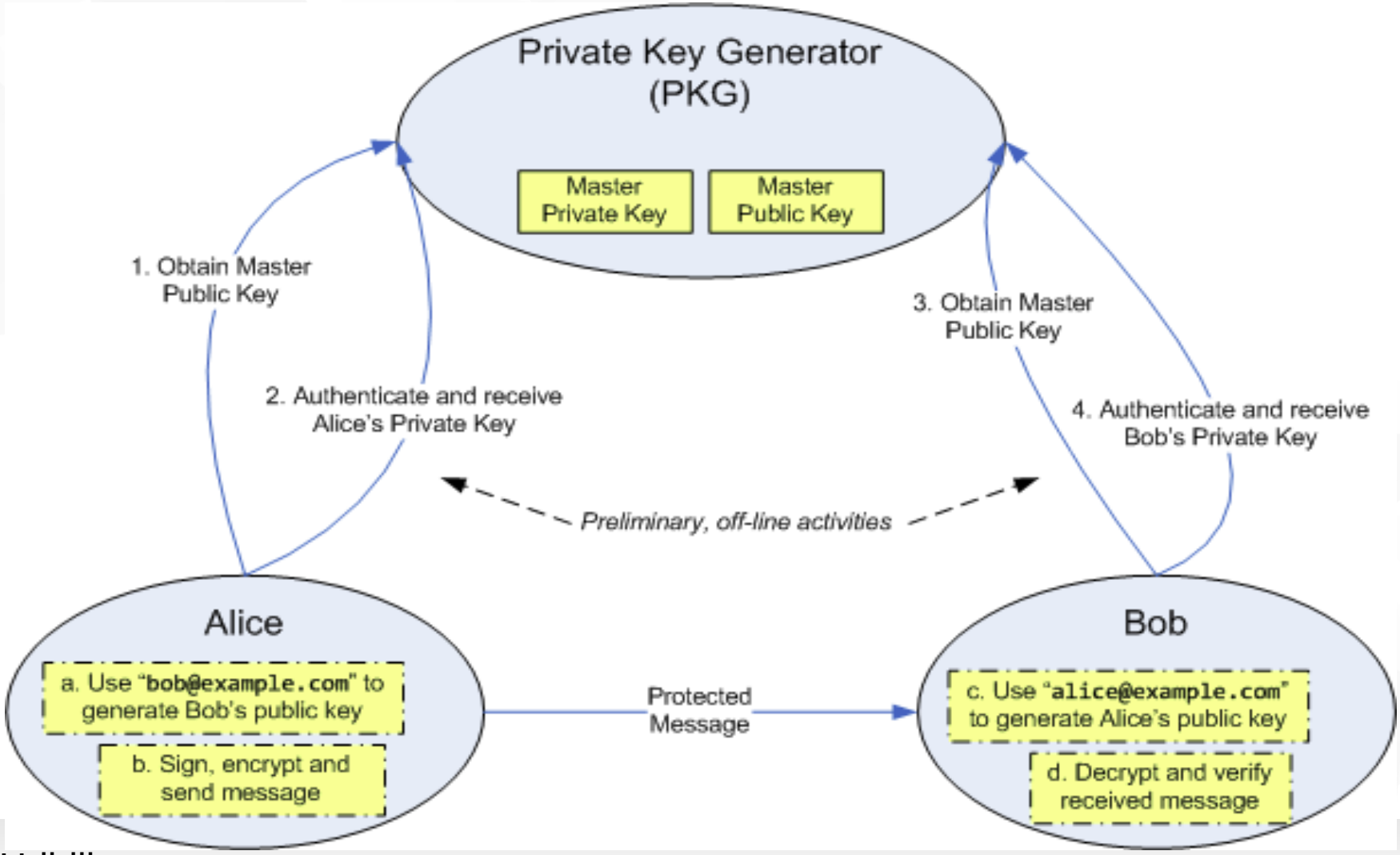


Identity Based Encryption

…… …

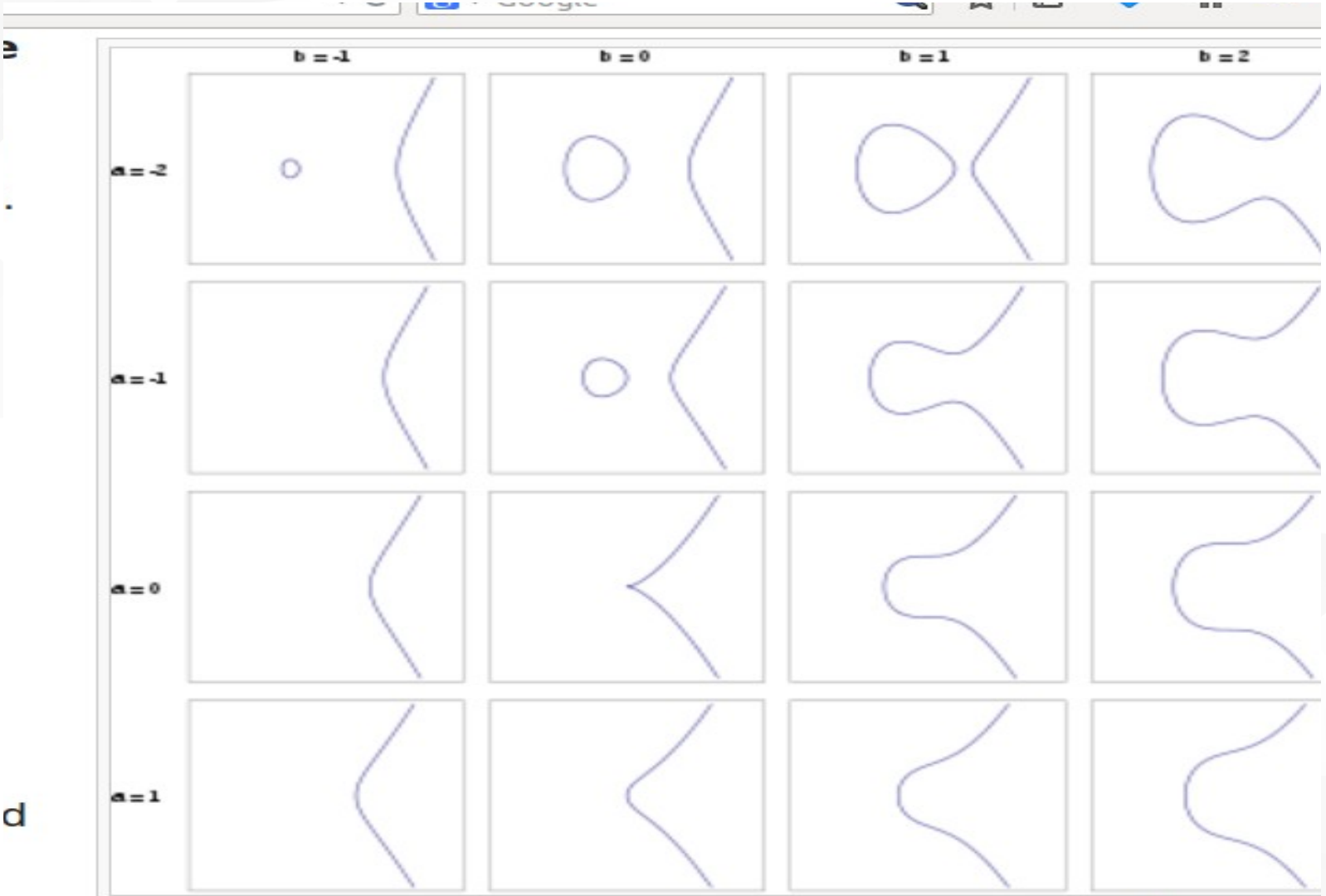
Identity Based Encryption

- Permettre d'utiliser une chaine arbitraire (email) comme clé publique
- Shéma efficace: utilise le bilinear pairing sur les courbes elliptiques



Elliptic curves

- Définie par $y^2=x^3+ax+b$



A catalog of elliptic curves. Region shown is $[-3,3]^2$ (For $a = 0$ and $b = 0$ the function is not smooth and therefore not a elliptic curve.)

IBE (Boneh–Franklin scheme)...

Setup

The Private Key Generator (PKG) chooses:

1. the public groups G_1 (with generator P) and G_2 as stated above, with the size of q depending on security parameter k,
2. the corresponding pairing e,
3. a random private master-key $K_m = s \in \mathbb{Z}_q^*$,
4. a public key $K_{pub} = sP$,
5. a public hash function $H_1 : \{0,1\}^* \to G_1^*$,
6. a public hash function $H_2 : G_2 \to \{0,1\}^n$ for some fixed n and
7. the message space and the cipher space $\mathcal{M} = \{0,1\}^n, \mathcal{C} = G_1^* \times \{0,1\}^n$

Extract

To create the public key for $ID \in \{0,1\}^*$, the PKG computes

1. $Q_{ID} = H_1(ID)$ and
2. the private key $d_{ID} = sQ_{ID}$ which is given to the user.

IBE (Boneh–Franklin scheme)...

Encrypt

Given $m \in \mathcal{M}$, the ciphertext c is obtained as follows:

1. $Q_{ID} = H_1(ID) \in G_1^*$,
2. choose random $r \in \mathbb{Z}_q^*$,
3. compute $g_{ID} = e(Q_{ID}, K_{pub}) \in G_2$ and
4. set $c = (rP, m \oplus H_2(g_{ID}^r))$.

Note that K_{pub} is the PKG's public key and thus independent of the recipient's ID.

Decrypt

Given $c = (u, v) \in \mathcal{C}$, the plaintext can be retrieved using the private key:

$$m = v \oplus H_2(e(d_{ID}, u))$$

Cryptographie à seuil

…… …

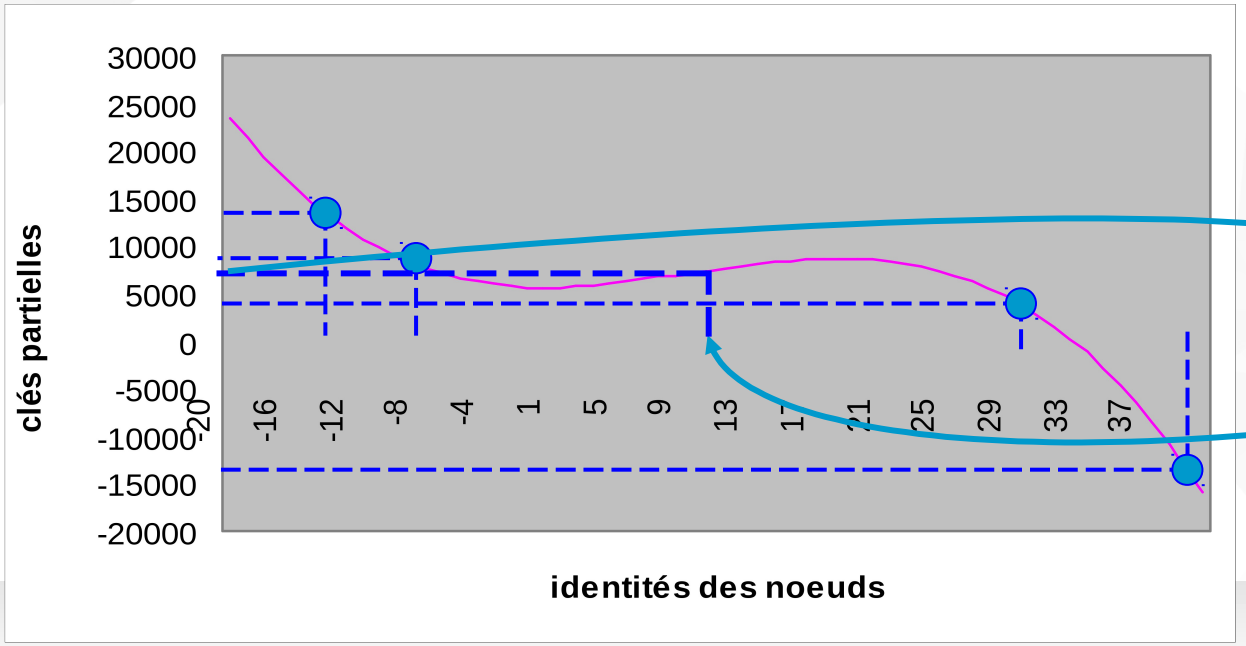
Partage de secret

- Un secret S est partagé entre N utilisateurs dont K (parmis N) peuvent le reconstruire
  - Exemple: Algorithme de Shamir,

$$f(x) = (S + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}) \bmod p$$

$$f(x) = \sum_{i=1}^{k} S_i Lid_i(x) \bmod p \qquad Lid_i(x) = \prod_{i=1, i!=j}^{k} \frac{x - id_j}{id_i - idj}$$

$$exemple: f(x) = (5555 + 25x + 26x^2 - x^3) \bmod 32099$$



$S_i = f(id_i)$

id_i

Cryptographie quantique

…… …

Cryptographie quantique

- Utiliser les propriétés de la physique quantique pour établir des protocoles de cryptographie
- Souvent utilisée pour transmettre des clés et non pas des messages
  - Les bits envoyés ne peuvent être qu'aléatoires
  - Tout esionnage d'une clé est détecté (la clé sera rejetée)
- Exemple:
  - Transmettre une clé secrète de A à B à distance d'une façon sécurisé en utilisant des objets quantiques et en garantissant la détection de tout espionnage (dans ce cas rejeter la clé)
- Après transmission de la clé, utiliser un algorithme de chiffrement classique

Annexe 1: détail du Data Encryption standard (DES)

…… …

DES (Matrice IP)

IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

IP^{-1}							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Table 7.2: DES initial permutation and inverse (IP and IP^{-1}).

Etapes du DES

- E: Expansion
  - Permutation avec expansion: entrée 32bits ➔ sortie 48 bits
- S: substitution
  - 8 substitutions (S_1, S_2,S_3,S_4,S_5,S_6) 6to4 bits
- P: permutation fixe de 32 bits

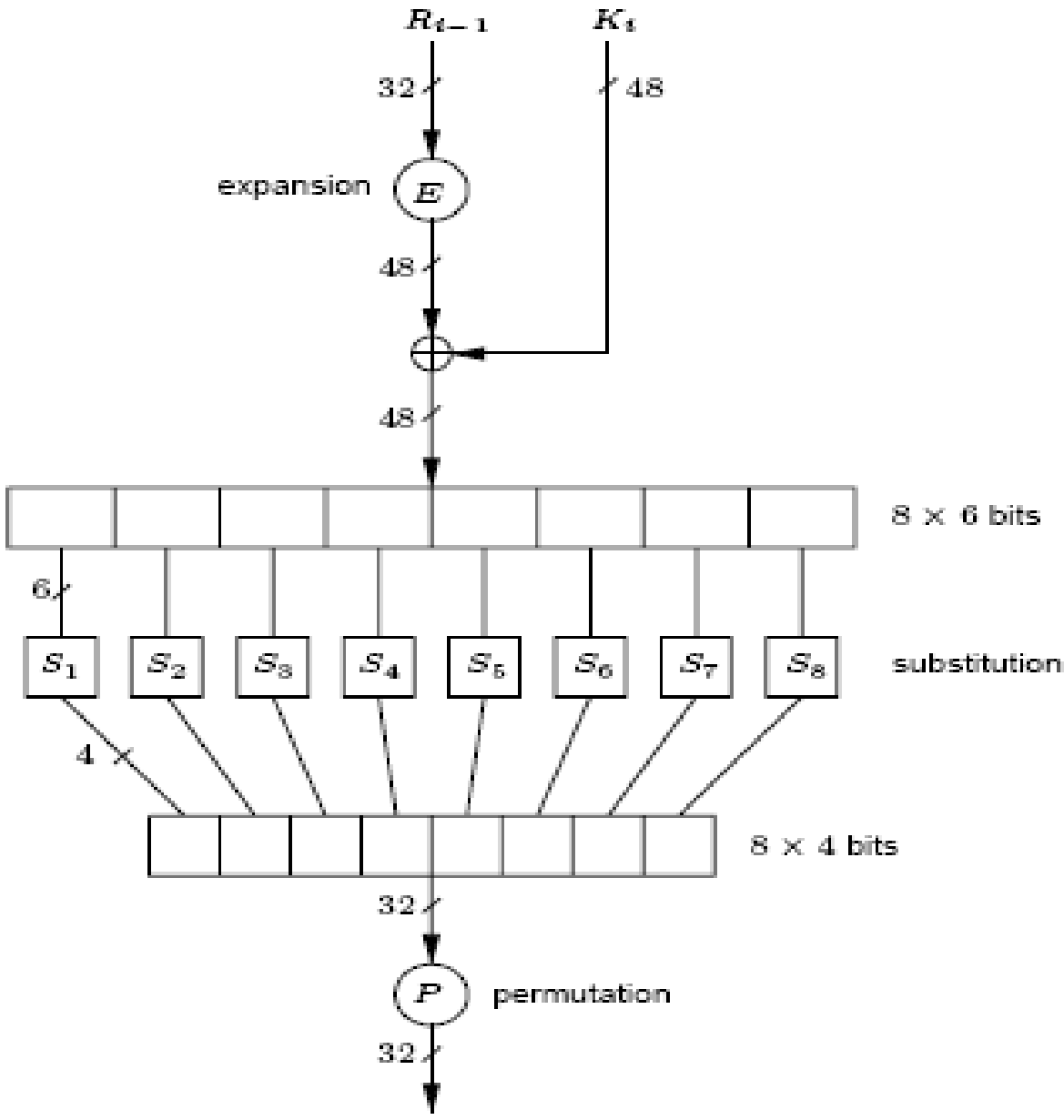


Figure 7.10: DES inner function f.

DES: Expansion et Permutation

Expand $R_{i-1} = r_1 r_2 \ldots r_{32}$ from 32 to 48 bits using E per Table 7.3: $T \leftarrow E(R_{i-1})$. (Thus $T = r_{32} r_1 r_2 \ldots r_{32} r_1$.)

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

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

Table 7.3: DES per-round functions: expansion E and permutation P.

$$L_i = R_{i-1};$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i), \text{ where } f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$$

Substitutions

(c) $T' \leftarrow (S_1(B_1), S_2(B_2), \ldots S_8(B_8))$. (Here $S_i(B_i)$ maps $B_i = b_1 b_2 \ldots b_6$ to the 4-bit entry in row r and column c of S_i in Table 7.8, page 260 where $r = 2 \cdot b_1 + b_6$, and $b_2 b_3 b_4 b_5$ is the radix-2 representation of $0 \leq c \leq 15$. Thus $S_1(011011)$ yields $r = 1, c = 13$, and output 5, i.e., binary 0101.)

$$L_i = R_{i-1};$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i), \text{ where } f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$$

DES s-boxes (substitutions)

row	column number															
	[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
S_2																
[0]	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
[1]	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
[2]	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
[3]	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3																
[0]	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
[1]	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
[2]	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
[3]	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4																
[0]	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
[1]	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
[2]	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
[3]	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

DES s-boxes (substitutions)

	S_5															
[0]	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
[1]	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
[2]	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
[3]	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
	S_6															
[0]	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
[1]	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
[2]	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
[3]	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
	S_7															
[0]	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
[1]	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
[2]	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
[3]	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
	S_8															
[0]	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
[1]	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
[2]	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
[3]	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Table 7.8: DES S-boxes.

DES: Calcul des sous-clés

7.83 Algorithm DES key schedule

INPUT: 64-bit key $K = k_1 \ldots k_{64}$ (including 8 odd-parity bits).
OUTPUT: sixteen 48-bit keys K_i, $1 \leq i \leq 16$.

1. Define v_i, $1 \leq i \leq 16$ as follows: $v_i = 1$ for $i \in \{1, 2, 9, 16\}$; $v_i = 2$ otherwise. (These are left-shift values for 28-bit circular rotations below.)
2. $T \leftarrow \text{PC1}(K)$; represent T as 28-bit halves (C_0, D_0). (Use PC1 in Table 7.4 to select bits from K: $C_0 = k_{57}k_{49}\ldots k_{36}$, $D_0 = k_{63}k_{55}\ldots k_4$.)
3. For i from 1 to 16, compute K_i as follows: $C_i \leftarrow (C_{i-1} \hookleftarrow v_i)$, $D_i \leftarrow (D_{i-1} \hookleftarrow v_i)$, $K_i \leftarrow \text{PC2}(C_i, D_i)$. (Use PC2 in Table 7.4 to select 48 bits from the concatenation $b_1 b_2 \ldots b_{56}$ of C_i and D_i: $K_i = b_{14}b_{17}\ldots b_{32}$. '$\hookleftarrow$' denotes left circular shift.)

PC1						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
above for C_i; below for D_i						
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

PC2					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Table 7.4: DES key schedule bit selections (PC1 and PC2).

Algorithme DES

Algorithm Data Encryption Standard (DES)

INPUT: plaintext $m_1 \ldots m_{64}$; 64-bit key $K = k_1 \ldots k_{64}$ (includes 8 parity bits).
OUTPUT: 64-bit ciphertext block $C = c_1 \ldots c_{64}$. (For decryption, see Note 7.84.)

1. (key schedule) Compute sixteen 48-bit round keys K_i from K using Algorithm 7.83.
2. $(L_0, R_0) \leftarrow \text{IP}(m_1 m_2 \ldots m_{64})$. (Use IP from Table 7.2 to permute bits; split the result into left and right 32-bit halves $L_0 = m_{58} m_{50} \ldots m_8$, $R_0 = m_{57} m_{49} \ldots m_7$.)
3. (16 rounds) for i from 1 to 16, compute L_i and R_i using Equations (7.4) and (7.5) above, computing $f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$ as follows:
   (a) Expand $R_{i-1} = r_1 r_2 \ldots r_{32}$ from 32 to 48 bits using E per Table 7.3: $T \leftarrow E(R_{i-1})$. (Thus $T = r_{32} r_1 r_2 \ldots r_{32} r_1$.)
   (b) $T' \leftarrow T \oplus K_i$. Represent T' as eight 6-bit character strings: $(B_1, \ldots, B_8) = T'$.
   (c) $T'' \leftarrow (S_1(B_1), S_2(B_2), \ldots S_8(B_8))$. (Here $S_i(B_i)$ maps $B_i = b_1 b_2 \ldots b_6$ to the 4-bit entry in row r and column c of S_i in Table 7.8, page 260 where $r = 2 \cdot b_1 + b_6$, and $b_2 b_3 b_4 b_5$ is the radix-2 representation of $0 \leq c \leq 15$. Thus $S_1(011011)$ yields $r = 1$, $c = 13$, and output 5, i.e., binary 0101.)
   (d) $T''' \leftarrow P(T'')$. (Use P per Table 7.3 to permute the 32 bits of $T'' = t_1 t_2 \ldots t_{32}$, yielding $t_{16} t_7 \ldots t_{25}$.)
4. $b_1 b_2 \ldots b_{64} \leftarrow (R_{16}, L_{16})$. (Exchange final blocks L_{16}, R_{16}.)
5. $C \leftarrow \text{IP}^{-1}(b_1 b_2 \ldots b_{64})$. (Transpose using IP^{-1} from Table 7.2; $C = b_{40} b_8 \ldots b_{25}$.)

- Exemple:
  - Pour mieux voir les permutations, considérons des chaines de caractères au lieu des bits. Les bits de parité sont représentés par des 0.

    abcdefg**0**hijklmn**0**opqrstu**0**vwxyzAB**0**CDEFGHI**0**JKLMNOP**0**QRS TUVW**0**XYZ1234**0**
  - Après la permutation PC1, on obtient C_0 et D_0 suivants

C_0							D_0						
X	Q	J	C	v	o	h	4	W	P	I	B	u	n
a	Y	R	K	D	w	p	G	3	V	O	H	A	t
i	b	Z	S	L	E	x	m	f	2	U	N	G	z
q	j	e	1	T	M	F	s	l	e	y	r	k	d

DES: calcul des sous clés

- La première clé K1 =PC2(C_1 , D_1)= iSDlQoCXbhqKcEwvMYZaFxpJtylVGdPAeUuz2sH4nrNml3Wb

i	S	D	l	Q	o
C	X	b	h	q	K
c	E	w	v	M	Y
Z	a	F	x	p	J
t	y	I	V	G	d
P	A	e	U	u	z
2	s	H	4	n	r
N	m	l	3	W	b

PC2					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

- La première clé K1 =PC2(C_1 , D_1)= iSDlQoCXbhqKcEwvMYZaFxpJtyIVGdPAeUuz2sH4nrNml3Wb

  ==> Les caractères R, L, j, T, g, O, f et k n’apparaissent pas dans K1

- La deuxième clés K2 sera:

  K2 = bLwTJhvQZajD1xpoFRSYXqiCmrBOz4ltyNnsUIAWgkGfePu

  ==> Les caractères absents dans K1 sont maintenant présents dans K2