Introduction à la cryptographie
I - À la recherche de la confidentialité

Carlos Aguilar

carlos.aguilar@enseeiht.fr

(INP ENSEEIHT, IRIT équipe IRT)

Références

Livres électroniques

Cornell,
https://www.cs.cornell.edu/courses/cs4830/2010fa/lecnotes.pdf
Bristol, http://www.cs.umd.edu/~waa/414-F11/IntroToCrypto.pdf
Stanford, http://crypto.stanford.edu/~dabo/cryptobook/

Cours fortement inspiré de :

[1] Stanford, https://www.coursera.org/course/crypto
[2] Univ. of Virginia, http://www.udacity.com/view#Course/cs387/

Plan

La cryptographie ...

C'est ...

- Un outil indispensable
- La base de nombreux protocoles de sécurité

Ce n'est pas ...

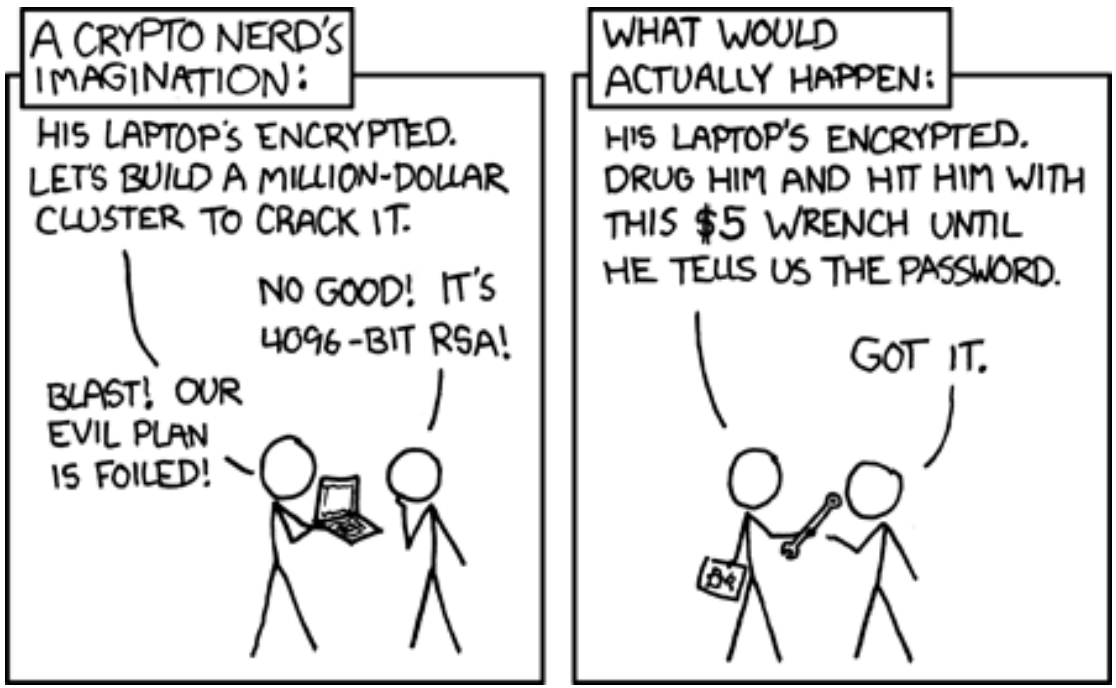
- La solution à tout problème
- Sûr sauf si implémenté correctement
- Sûr sauf si utilisé correctement

Et surtout ...

Ce n'est pas forcément incompréhensible ! (ça peut même être rigolo, sisi)

La cryptographie ...

http://www.xkcd.com



Cryptologie et cryptographie

Définitions

La Cryptologie est “la science du secret”, elle est composée de :

- la cryptographie (écriture de secrets)
- la cryptanalyse (recherche des secrets)

Un petit quiz [2] : Quelles actions ont un rapport avec la cryptologie ?

1. Ouvrir une porte
2. Jouer au poker
3. Se connecter à un compte (sur le net)
4. Faire une recherche avec Google

Cryptologie et cryptographie

Définitions

La Cryptologie est “la science du secret”, elle est composée de :

- la cryptographie (écriture de secrets)
- la cryptanalyse (recherche des secrets)

Un petit quiz [2] : Quelles actions ont un rapport avec la cryptologie ?

1. Ouvrir une porte
2. Jouer au poker
3. Se connecter à un compte (sur le net)
4. Faire une recherche avec Google

Complexité et cryptographie (1/2)

Bits de sécurité et attaques

Système avec k bits de sécurité $\Leftrightarrow$ Meilleure attaque en 2^k operations

“Opérations” n’est pas bien défini... supposé être proche d’un cycle d’horloge

Repères

Pour différentes valeurs de k, combien de temps il faut pour faire 2^k opérations ?

- $k = 30 \rightarrow$ 1 seconde sur un ordinateur
- $k = 60 \rightarrow$ 1 seconde de puissance de calcul sur terre (PCST)
- $k = 85 \rightarrow$ 1 an de PCST
- $k = 100 \rightarrow$ 32000 ans de PCST
- $k = 115 \rightarrow$ l’age de l’univers de PCST

Sécurités considérées aujourd’hui $k = 100, 112, 128, 256$

Complexité et cryptographie (2/2)

Principe

- Combien de nombres de 1 bit il y a ?

Les grands espaces de nombres

Avec très peu de bits k on peut écrire des nombres impossibles à deviner
Repère : l'humanité a "écrit" moins de 2^{100} bits dans son histoire

Combien de bits il faut "écrire" en moyenne pour tomber sur un nombre de 128 bits en particulier (e.g. deviner une combinaison de 128 bits ouvrant un coffre) ?

Complexité et cryptographie (2/2)

Principe

- Combien de nombres de 1 bit il y a ?
- Combien de nombres de 2 bits il y a ?

Les grands espaces de nombres

Avec très peu de bits k on peut écrire des nombres impossibles à deviner
Repère : l'humanité a "écrit" moins de 2^{100} bits dans son histoire

Combien de bits il faut "écrire" en moyenne pour tomber sur un nombre de 128 bits en particulier (e.g. deviner une combinaison de 128 bits ouvrant un coffre) ?

Complexité et cryptographie (2/2)

Principe

- Combien de nombres de 1 bit il y a ?
- Combien de nombres de 2 bits il y a ?

$\rightarrow$ Pour un bloc de k bits il y a 2^k possibilités

Les grands espaces de nombres

Avec très peu de bits k on peut écrire des nombres impossibles à deviner
Repère : l'humanité a "écrit" moins de 2^{100} bits dans son histoire

Combien de bits il faut "écrire" en moyenne pour tomber sur un nombre de 128 bits en particulier (e.g. deviner une combinaison de 128 bits ouvrant un coffre) ?

Complexité et cryptographie (2/2)

Principe

- Combien de nombres de 1 bit il y a ?
- Combien de nombres de 2 bits il y a ?

$\rightarrow$ Pour un bloc de k bits il y a 2^k possibilités

Les grands espaces de nombres

Avec très peu de bits k on peut écrire des nombres impossibles à deviner
Repère : l'humanité a "écrit" moins de 2^{100} bits dans son histoire

Combien de bits il faut "écrire" en moyenne pour tomber sur un nombre de 128 bits en particulier (e.g. deviner une combinaison de 128 bits ouvrant un coffre) ?
Il faut environ 2^{128} tests de 128 bits $\rightarrow 128 * 2^{128}$ bits

Plan

Définitions du chiffrement

Termes français

- Clair : message non chiffré $m \in \mathcal{M}$
- Chiffré : message chiffré $c \in \mathcal{C}$
- Clé symétrique : secret permettant de chiffrer/déchiffrer $sk \in \mathcal{K}$
- Chiffrement : passage d'un clair à un chiffré $c \leftarrow Enc(sk, m)$
- Déchiffrement : passage d'un chiffré à un clair $m \leftarrow Dec(sk, c)$
- Crypter/Décrypter : a proscrire
- Clé secrète : ambigu

Termes anglais

Plaintext, ciphertext, symetric key, encryption, decryption

Le chiffrement

Utilité et limites

Permet de cacher des informations (échangées ou stockées).

- N'occulte pas la présence/taille/timing des données
- N'évite pas les modifications/rejeus
- On ne peut pas tout chiffrer (par ex. infos de routage)

Principe de Kerckhoff (1883) sur le secret des algorithmes

"Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi"

Ne réside pas dans l'algorithme utilisé mais dans une de ses entrées : la clé

Algos symétriques : une seule clé (secrète) pour chiffrer et déchiffrer
Algos asymétriques : clé publique pour chiffrer, secrète pour déchiffrer

Un autre principe : la (bi-)clé doit être petite !

Symétrique vous dites ?

Une même clé pour chiffrer et déchiffrer

Alice veut envoyer un message à Bob

- Alice chiffre le clair en utilisant une clé sk et obtient un chiffré
- Alice envoie le chiffré ou le met dans un endroit où Bob peut le récupérer (par ex. sur le frigo)
- Bob utilise **la même clé** sk pour déchiffrer le message reçu

Plus formellement

- Chiffrement d'un clair m : $c = Enc(sk, m)$
- Déchiffrement d'un chiffré c : $m = Dec(sk, c)$
- Consistance : $\forall m, sk : Dec(sk, Enc(sk, m)) = m$
- Sécurité : les chiffrés ne dévoilent rien sur les messages ou la clé

Un autre quiz [2]

Lequel de ces chiffrements symétriques est consistant ?

$\mathcal{M} = \mathcal{C} = \mathcal{K} = \{1, 2, 3, \ldots\}$

1. $Enc(k, m) = k + m$, $Dec(k, c) = c - k$
2. $Enc(k, m) = m$, $Dec(k, c) = c$
3. $Enc(k, m) = m\%k$, $Dec(k, c) = c * k$

Un autre quiz [2]

Lequel de ces chiffrements symétriques est consistant ?

$\mathcal{M} = \mathcal{C} = \mathcal{K} = \{1, 2, 3, \ldots\}$

1. $Enc(k, m) = k + m$, $Dec(k, c) = c - k$
2. $Enc(k, m) = m$, $Dec(k, c) = c$
3. $Enc(k, m) = m\%k$, $Dec(k, c) = c * k$

Plan

XOR à la rescousse

Notation

$\oplus$ = XOR = “OU-exclusif” = “Somme modulo 2” :
$0 \oplus 0 = 1 \oplus 1 = 0$, $0 \oplus 1 = 1 \oplus 0 = 1$

Quiz [2] : Que vaut $x \oplus y \oplus x$?

1. 0
2. x
3. y
4. Ça dépend de x

Comment l’appliquer à la cryptographie ?

XOR à la rescousse

Notation

$\oplus$ = XOR = “OU-exclusif” = “Somme modulo 2” :
$0 \oplus 0 = 1 \oplus 1 = 0$, $0 \oplus 1 = 1 \oplus 0 = 1$

Quiz [2] : Que vaut $x \oplus y \oplus x$?

1. 0
2. x
3. y
4. Ça dépend de x

Comment l’appliquer à la cryptographie ?

Le One Time Pad (Vernam, 1917)

Un chiffrement symétrique parfaitement sûr et simple

message $\oplus$ clé = chiffré chiffré $\oplus$ clé = message

```
message : 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1
clé     : 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0
=================================
chiffré : 1 0 0 1 1 0 1 1 0 0 0 1 1 1 1
```

Preuve

X = message : généré par un utilisateur ou un algorithme
Y = clé : suite de bits unif. aléatoires indépendante de X
Z = chiffré : obtenu par l'algorithme suivant

Z suit une loi uniforme.

$p(Z = z) = \sum_x p(X = x \cap Y = x \oplus z) = \sum_x p(X = x)p(Y = x \oplus z)$
$p(Z = z) = 1/2^n \sum_x p(X = x) = 1/2^n$

Le One Time Pad (Vernam, 1917)

Un chiffrement symétrique parfaitement sûr et simple

message $\oplus$ clé = chiffré chiffré $\oplus$ clé = message

```
message : 0 1
clé     : 1 1
=====================================
chiffré : 1 0
```

Preuve

X = message : généré par un utilisateur ou un algorithme
Y = clé : suite de bits unif. aléatoires indépendante de X
Z = chiffré : obtenu par l'algorithme suivant

Z suit une loi uniforme.

$p(Z = z) = \sum_x p(X = x \cap Y = x \oplus z) = \sum_x p(X = x)p(Y = x \oplus z)$
$p(Z = z) = 1/2^n \sum_x p(X = x) = 1/2^n$

Le One Time Pad (Vernam, 1917)

Un chiffrement symétrique parfaitement sûr et simple

message $\oplus$ clé = chiffré chiffré $\oplus$ clé = message

```
message : X X
clé     : X X
=====================================
chiffré : 1 0
```

Preuve

X = message : généré par un utilisateur ou un algorithme
Y = clé : suite de bits unif. aléatoires indépendante de X
Z = chiffré : obtenu par l'algorithme suivant

Z suit une loi uniforme.
$p(Z = z) = \sum_x p(X = x \cap Y = x \oplus z) = \sum_x p(X = x)p(Y = x \oplus z)$
$p(Z = z) = 1/2^n \sum_x p(X = x) = 1/2^n$

Le One Time Pad (Vernam, 1917)

Un chiffrement symétrique parfaitement sûr et simple

message $\oplus$ clé = chiffré chiffré $\oplus$ clé = message

```
message : X X      0 0 | 0 1 | 1 0 | 1 1
clé     : X X      1 0 | 1 1 | 0 0 | 0 1
=================================================
chiffré : 1 0      1 0 | 1 0 | 1 0 | 1 0
```

Preuve

X = message : généré par un utilisateur ou un algorithme
Y = clé : suite de bits unif. aléatoires indépendante de X
Z = chiffré : obtenu par l'algorithme suivant

Z suit une loi uniforme.

$p(Z = z) = \sum_x p(X = x \cap Y = x \oplus z) = \sum_x p(X = x)p(Y = x \oplus z)$
$p(Z = z) = 1/2^n \sum_x p(X = x) = 1/2^n$

Le One Time Pad (Vernam, 1917)

Un chiffrement symétrique parfaitement sûr et simple

message $\oplus$ clé = chiffré chiffré $\oplus$ clé = message

```
message : 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1
clé     : 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0
=================================
chiffré : 1 0 0 1 1 0 1 1 0 0 0 1 1 1 1
```

Preuve

X = message : généré par un utilisateur ou un algorithme
Y = clé : suite de bits unif. aléatoires indépendante de X
Z = chiffré : obtenu par l'algorithme suivant

Z suit une loi uniforme.

$p(Z = z) = \sum_x p(X = x \cap Y = x \oplus z) = \sum_x p(X = x)p(Y = x \oplus z)$
$p(Z = z) = 1/2^n \sum_x p(X = x) = 1/2^n$

Parfaitement sûr et simple ... mais peu pratique

Problèmes de Malléabilité

Si un attaquant voit passer un chiffré c du clair m et il le modifie en $c' = c \oplus x$ alors c' aura pour clair associé $m' = m \oplus x$

Problèmes de gestion de clé

Clé aussi grande que le message et non réutilisable ! !
$c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2$

Les mauvaises nouvelles

Théorème de Shanon

Si un système de chiffrement est parfaitement sûr, alors $|K| \geq |M|$

Preuve

Supposons par l'absurde que $|K| < |M|$ et qu'un attaquant étudie $Y = c$
Si il calcule M_0 l'ensemble des messages obtenus par $Dec(k, c)$ pour $k \in K$

n sait que le message associé à c est dans M_0 et $|M_0| = |K| < |M|$
$\Rightarrow$ On a appris quelque chose sur m en étudiant c

OTP : Malléabilité [2]

Contexte

Alice et Bob sont d'accord sur une clé, ils veulent s'échanger un message "O"ou "N" (pour oui et non) en utilisant un OTP. Eve sait que le clair sera un "O"ou un "N", intercepte le chiffré c et souhaite modifier le message de façon que si c'était un chiffré de "O" ça devienne un chiffré de "N" et inversement

Comment doit-elle modifier c avant de le renvoyer ?

Piste : $\text{“}O\text{”} = 01001111$ et $\text{“}N\text{”} = 01001110$

OTP : Malléabilité [2]

Contexte

Alice et Bob sont d'accord sur une clé, ils veulent s'échanger un message "O"ou "N" (pour oui et non) en utilisant un OTP. Eve sait que le clair sera un "O"ou un "N", intercepte le chiffré c et souhaite modifier le message de façon que si c'était un chiffré de "O" ça devienne un chiffré de "N" et inversement

Comment doit-elle modifier c avant de le renvoyer ?

Piste : $\text{“}O\text{”} = 01001111$ et $\text{“}N\text{”} = 01001110$

$c' = c \oplus (\text{“}O\text{”} \oplus \text{“}N\text{”}) = c \oplus 01001111 \oplus 01001110 = c \oplus 00000001$

OTP : Partage de secrets [2]

Partage à deux

Alice à un secret x de k bits qui permet de lancer des missiles nucléaires
Il génère y de k bits indépendamment et uniformément au hasard et donne :

- $x \oplus y$ à sa maman
- y à son papa

Maman et Papa pourront lancer la guerre nucléaire mais il faudra qu'il s'y mettent à deux

Quelles affirmations sont vraies ?

1. y suit une distribution uniforme (entiers de k bits)
2. $x \oplus y$ suit une distribution uniforme (entiers de k bits
3. $(y, x \oplus y)$ suit une distribution uniforme sur (couples de k bits)

OTP : Partage de secrets [2]

Partage à deux

Alice à un secret x de k bits qui permet de lancer des missiles nucléaires
Il génère y de k bits indépendamment et uniformément au hasard et donne :

- $x \oplus y$ à sa maman
- y à son papa

Maman et Papa pourront lancer la guerre nucléaire mais il faudra qu'il s'y mettent à deux

Quelles affirmations sont vraies ?

1. y suit une distribution uniforme (entiers de k bits)
2. $x \oplus y$ suit une distribution uniforme (entiers de k bits
3. $(y, x \oplus y)$ suit une distribution uniforme sur (couples de k bits)

OTP : Partage de secrets [2]

Partage à deux

Alice à un secret x de k bits qui permet de lancer des missiles nucléaires
Il génère y de k bits indépendamment et uniformément au hasard et donne :

- $x \oplus y$ à sa maman
- y à son papa

Maman et Papa pourront lancer la guerre nucléaire mais il faudra qu'il s'y mettent à deux

Alice veut faire un partage à 4

Comment peut elle définir les secrets partagés ?

OTP : Partage de secrets [2]

Partage à deux

Alice à un secret x de k bits qui permet de lancer des missiles nucléaires
Il génère y de k bits indépendamment et uniformément au hasard et donne :

- $x \oplus y$ à sa maman
- y à son papa

Maman et Papa pourront lancer la guerre nucléaire mais il faudra qu'il s'y mettent à deux

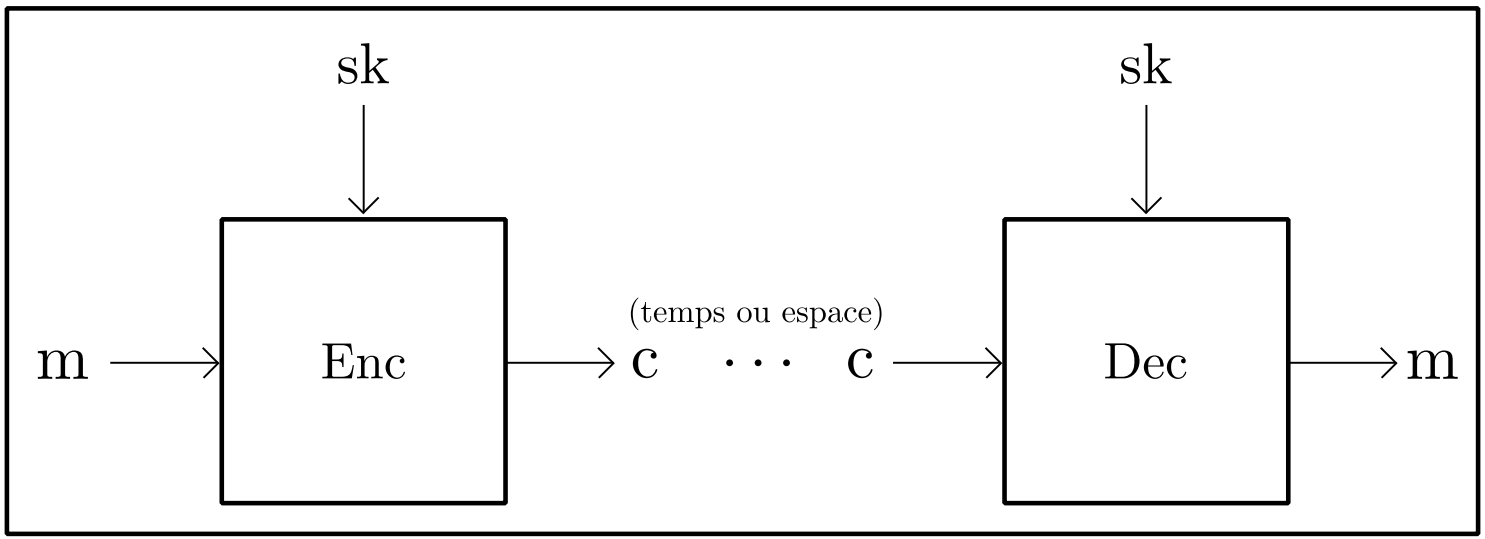
Alice veut faire un partage à 4

Comment peut elle définir les secrets partagés ?

$s_1 = x \oplus y_1, s_2 = y_1 \oplus y_2, s_3 = y_2 \oplus y_3, s_4 = y_3$

Plan

Les algorithmes de chiffrement symétrique



Tailles

On aimerait que la taille de la clé aie un lien avec la sécurité mais ne limite pas la quantité de données que l'on peut chiffrer

Chiffrement à flots (1/2)

Idée : reprendre l'algo du OTP

message $\oplus$ masque = chiffré chiffré $\oplus$ masque = message

... en remplaçant le masque aléatoire par un masque pseudoaléatoire

Générateur pseudo aléatoire

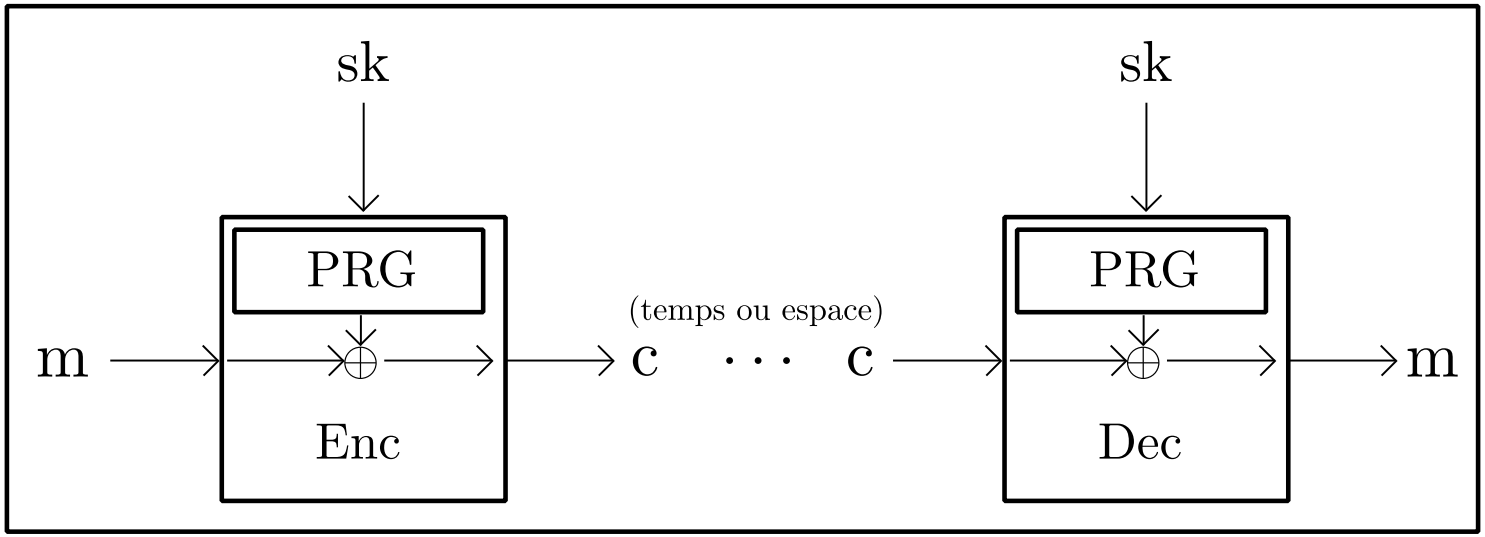
Fonction : $\mathrm{PRG} : \{0,1\}^s \longrightarrow \{0,1\}^n$ avec $n >> s$ avec
Sortie de PRG rapidement calculable (bit par bit)
$\forall s\ \mathrm{PRG}(s)$ a l'air aléatoire

Utilisation

Très utilisé en télécommunications :

- le flot généré bit par bit permet de chiffrer/déchiffrer une communication au fur et à mesure simplement
- la chaîne pseudoaléatoire peut être pre-générée pour éviter d'introduire des délais

Chiffrement à flots (2/2)



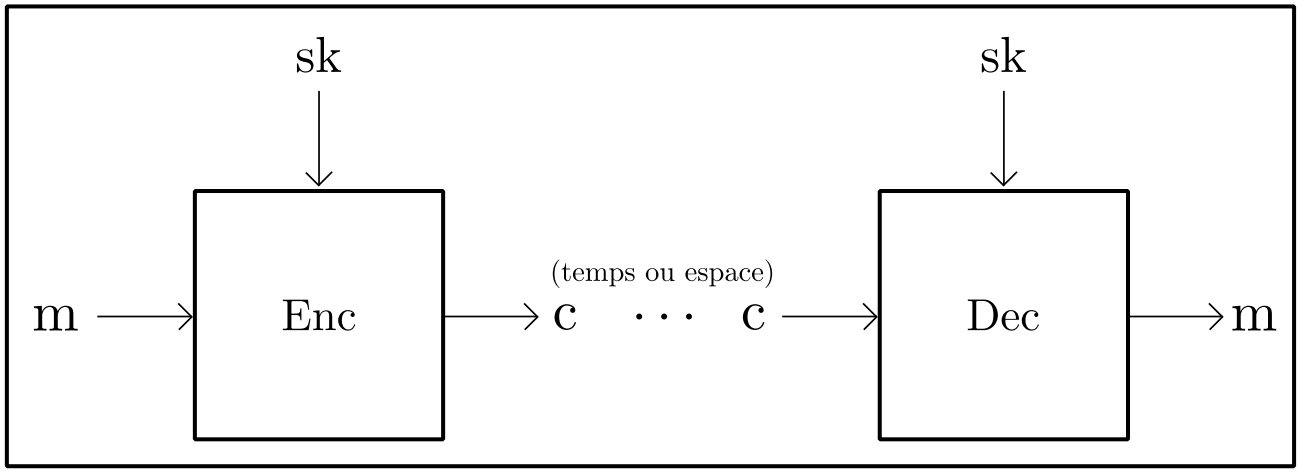
Il faut faire attention à la synchro !

Exemples

RC4 (utilisé dans WEP et WPA TKIP)
Réseaux cellulaires : A5/1-3 (GSM/GPRS), SNOW(3G) (UMTS), KASUMI (LTE)
Standards crypto (Projet eSTREAM) : Salsa20, SOSEMANUK (Fr)

Tailles de clés : 128/192/256 bits, Vitesse de chiffrement/déchiffrement : 1-10Gbits/s

Le chiffrement par blocs

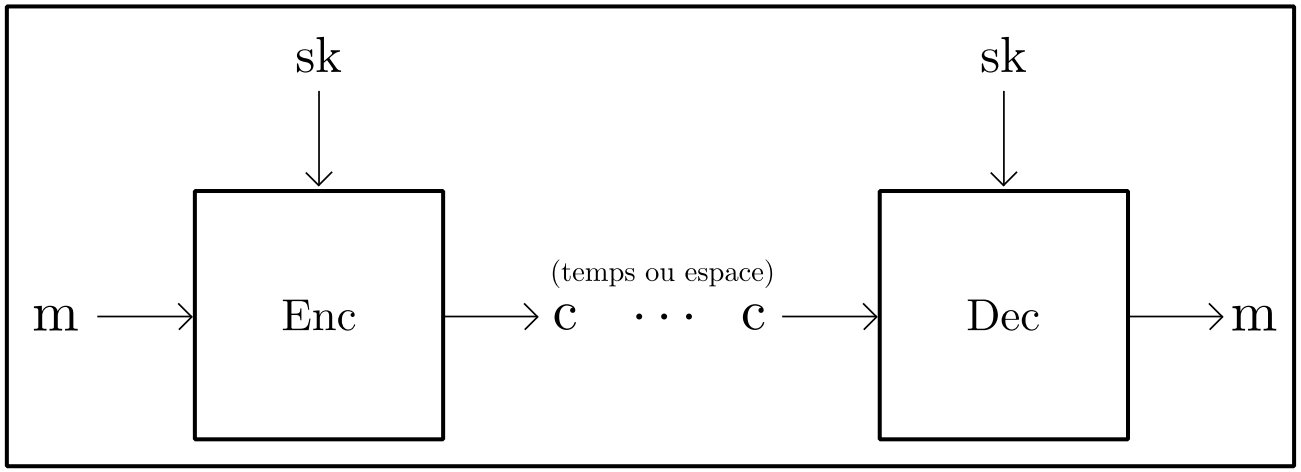


Symétrique : une même clé pour chiffrer et déchiffrer

Alice veut envoyer un message à Bob

- Alice chiffre le clair en utilisant une clé sk et obtient un chiffré
- Alice envoie le chiffré ou le met dans un endroit où Bob peut le récupérer
- Bob utilise **la même clé** sk pour déchiffrer le message reçu

Le chiffrement par blocs



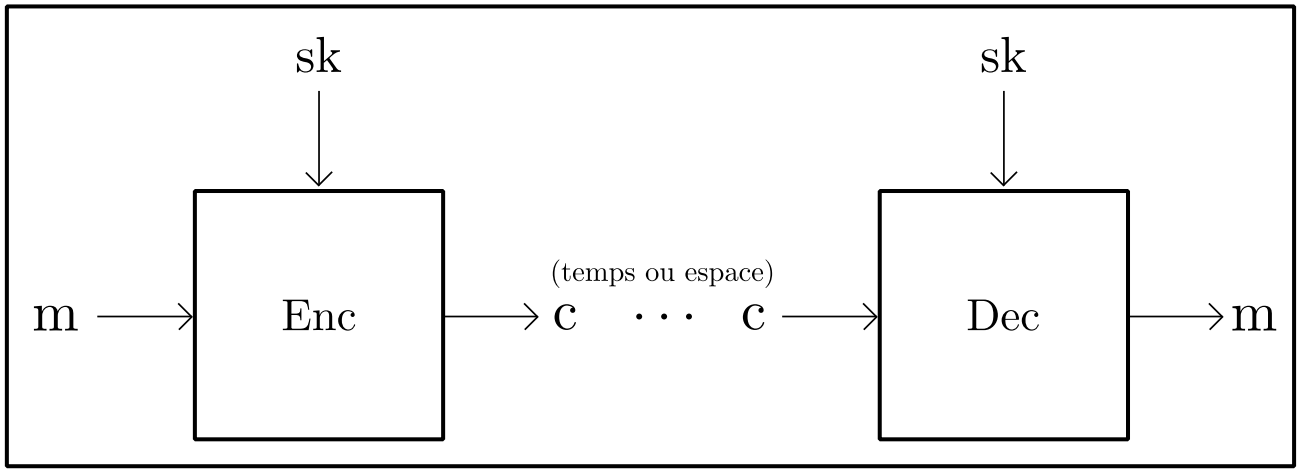
Le chiffrement par blocs

Clé de k bits (typiquement 128 ou 256)
On chiffre les données par blocs de bits (généralement k aussi)

- Chaque bloc de k bits du message donne un chiffré de k bits
- Enc fait l'opération dans un sens et Dec dans l'autre

Il existe une autre approche dite par flots

Le chiffrement par blocs



Comment c'est construit

C'est une machine à laver qui fait des opérations réversibles

- Clé de chiffrement utilisée pour générer plusieurs sous-clés, une par tour
- On itère sur le bloc d'entrée sur plusieurs tours
- À chaque tour on mélange les bits du message et de la sous-clé associée entre eux

Chaque opération est refaite dans l'autre sens lors du déchiffrement
Animation extérieure tirée de `http://www.formaestudio.com/rijndaelinspector`

Algorithmes à connaître (1/2)

Data Encryption Standard (DES, 1975)

Clés de 64 bits mais un bit par octet de parité $\rightarrow$ 56 bits
Coût attaques :

- Annonces théoriques 1977 (20M$), 1993 (1M$)
- 1998 (250K$, 2 jours), pratique réalisé par la EFF
- 2008 (150K$, 7 heures), pratique machine composée de 128 FPGAs (en vente)

Triple DES (3DES)

Chiffrement $\mathrm{Enc}(k_1) \circ \mathrm{Dec}(k_2) \circ \mathrm{Enc}(k_3)$ $\rightarrow$ Déchiffrement $\mathrm{Dec}(k_3) \circ \mathrm{Enc}(k_2) \circ \mathrm{Dec}(k_1)$
Attaque Meet-in-the-Middle (partant d'un couple clair=m/chiffré=c)
- Stocker 2^{56} chiffrés de m
- Déchiffrer c partiellement (un Dec un Enc) 2^{112} fois
- Chercher à chaque fois si le déchiffré partiel est dans la table de chiffrés
$\rightarrow$ 112 bits de sécurité

Pourquoi pas faire un Double DES ?

Algorithmes à connaître (1/2)

Data Encryption Standard (DES, 1975)

Clés de 64 bits mais un bit par octet de parité $\rightarrow$ 56 bits
Coût attaques :

- Annonces théoriques 1977 (20M$), 1993 (1M$)
- 1998 (250K$, 2 jours), pratique réalisé par la EFF
- 2008 (150K$, 7 heures), pratique machine composée de 128 FPGAs (en vente)

Triple DES (3DES)

Chiffrement $\mathrm{Enc}(k_1) \circ \mathrm{Dec}(k_2) \circ \mathrm{Enc}(k_3)$ $\rightarrow$ Déchiffrement $\mathrm{Dec}(k_3) \circ \mathrm{Enc}(k_2) \circ \mathrm{Dec}(k_1)$
Attaque Meet-in-the-Middle (partant d'un couple clair=m/chiffré=c)
- Stocker 2^{56} chiffrés de m
- Déchiffrer c partiellement (un Dec un Enc) 2^{112} fois
- Chercher à chaque fois si le déchiffré partiel est dans la table de chiffrés
$\rightarrow$ 112 bits de sécurité

Pourquoi pas faire un Double DES ?

Meet-in-the-Middle $\rightarrow$ 56 bits de sécurité

Algorithmes à connaître (2/2)

Standard Actuel : Advanced Encryption Standard (AES)

- Concours ouvert en 1997 (pas comme pour DES), choix final en 2001
- Organisé par le National Institute of Standards and Technology (NIST)
- 15 soumission $\rightarrow$ 5 finalistes $\rightarrow$ Rjindael

Tailles

- Taille de bloc (clair/chiffré) fixe de 128 bits
- Tailles de clés : 128bits, 192bits, ou 256bits
- Meilleure attaque connue en 2^{k-2} (attaque biclique 2011)

Vitesse

Sur mon ordinateur portable : 4Gbits/s
Saturation de ma fibre (100Mbits/s) avec du trafic chiffré : % de mon CPU

Algorithmes à connaître (2/2)

Standard Actuel : Advanced Encryption Standard (AES)

- Concours ouvert en 1997 (pas comme pour DES), choix final en 2001
- Organisé par le National Institute of Standards and Technology (NIST)
- 15 soumission $\rightarrow$ 5 finalistes $\rightarrow$ Rjindael

Tailles

- Taille de bloc (clair/chiffré) fixe de 128 bits
- Tailles de clés : 128bits, 192bits, ou 256bits
- Meilleure attaque connue en 2^{k-2} (attaque biclique 2011)

Vitesse

Sur mon ordinateur portable : 4Gbits/s
Saturation de ma fibre (100Mbits/s) avec du trafic chiffré : 2.5% de mon CPU

Modes

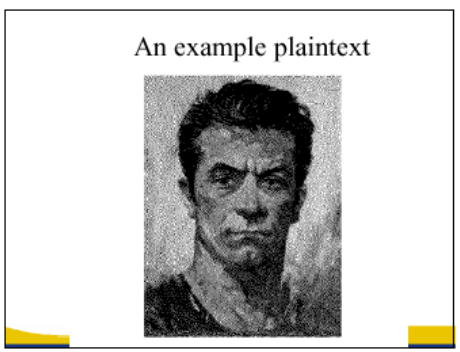
Utilisation

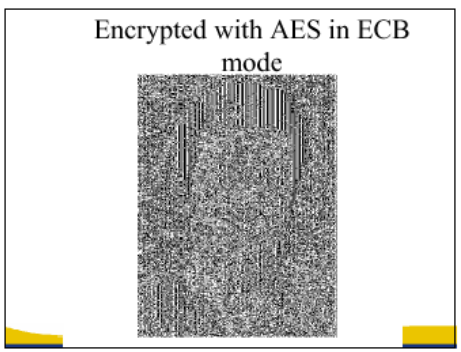
Pour un algorithme il faut choisir une taille de clé et un mode de fonctionnement

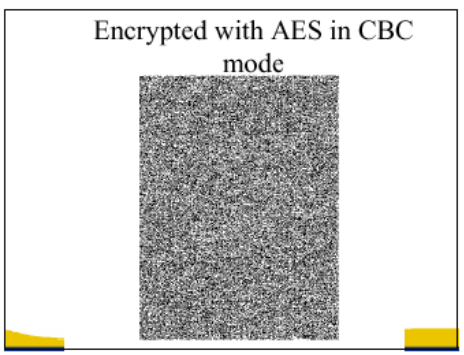
Les modes de fonctionnement

Pour une clé secrète donnée on a un chiffrement déterministe
Mode ECB : deux fois le même clair $\rightarrow$ deux fois le même chiffré

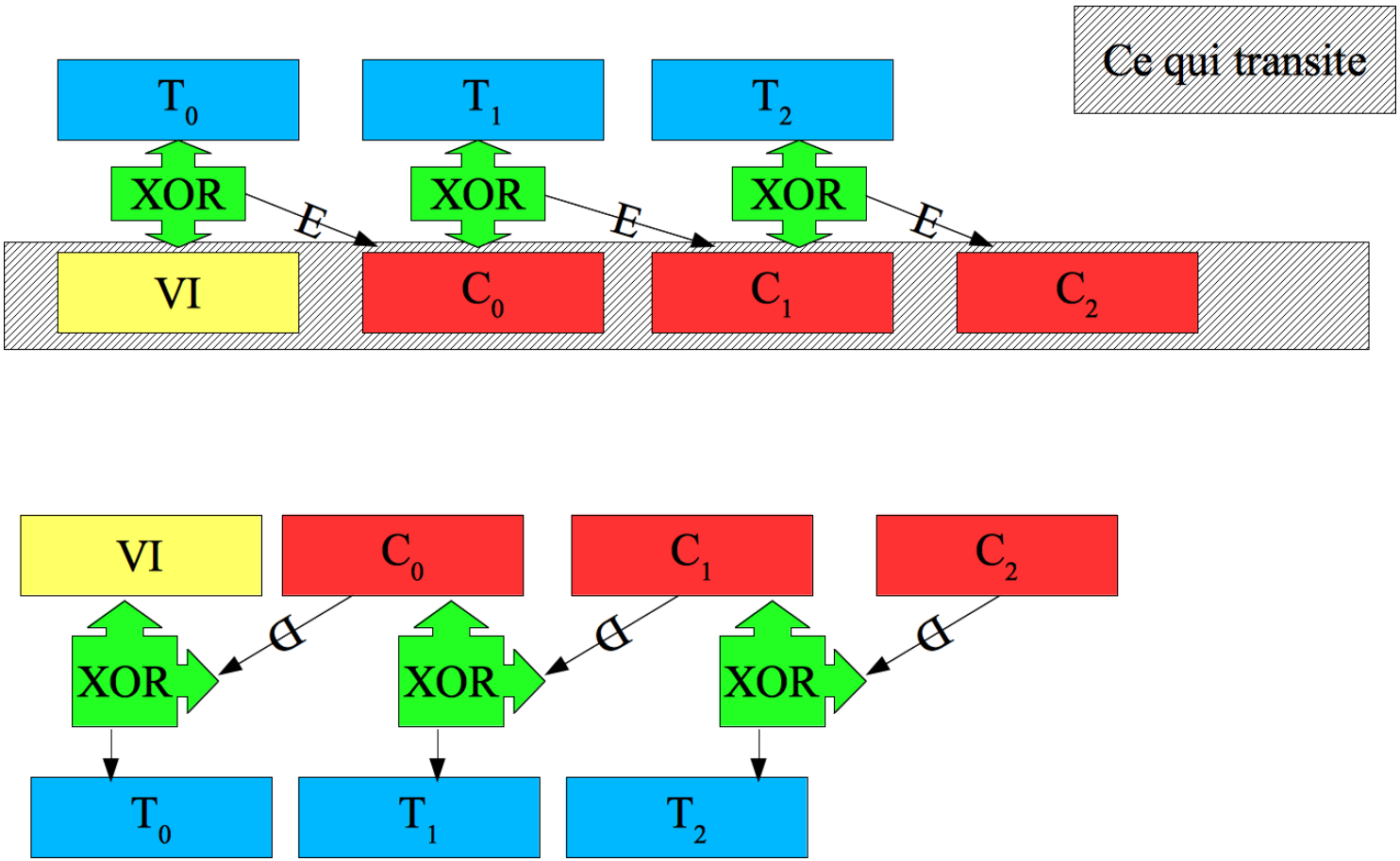
Modes sûrs CBC, CTR, etc.





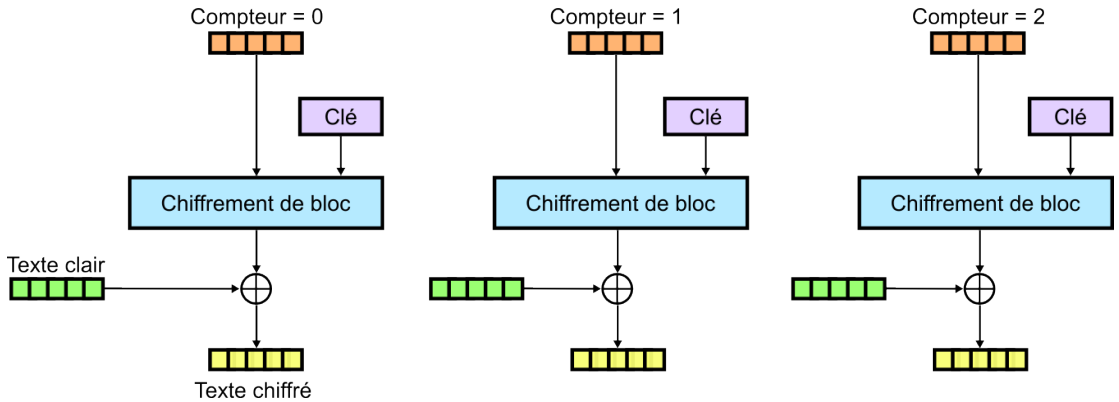


Le mode Cipher-Block Chain (CBC)



Le mode compteur (CTR)

Principe



Source : `https://fr.wikipedia.org/wiki/Mode_doperation_(cryptographie)`

Propriétés

AES chiffrement par blocs sûr $\Rightarrow$ AES-CTR chiffrement à flots sûr
Parallélisable au chiffrement/déchiffrement, accès aléatoire

Le problème de l'échange de clés

Problème

Si A et B connaissent une clé symétrique commune *sk* ils peuvent utiliser un algorithme de chiffrement symétrique pour chiffrer leurs échanges mais...

Comment font ils pour obtenir cette clé commune ? ? ? ?

Canal sûr

A donne à B une clé USB en main
Distance : Iriez-vous en Chine pour faire ça ?
Grands groupes : Imaginez la queue devant Google...

Et si...

- Pour chiffrer les messages envoyés à Google on utilisait une clé publique *pub*
- Pour déchiffrer ces messages il fallait une **autre** clé *priv*, dite privée
- Tout le monde connaissait *pub* (e.g. installée dans les navigateurs)

Que pourrions nous faire ? (piste : on veut avoir une clé symétrique commune *sk*)

Le problème de l'échange de clés

Problème

Si A et B connaissent une clé symétrique commune *sk* ils peuvent utiliser un algorithme de chiffrement symétrique pour chiffrer leurs échanges mais...

Comment font ils pour obtenir cette clé commune ? ? ? ?

Canal sûr

A donne à B une clé USB en main
Distance : Iriez-vous en Chine pour faire ça ?
Grands groupes : Imaginez la queue devant Google...

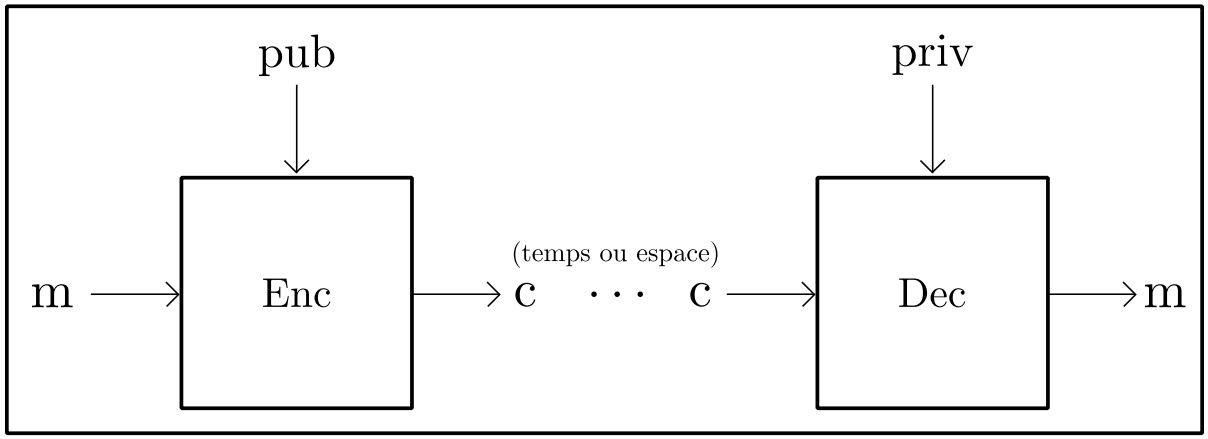
Et si...

- Pour chiffrer les messages envoyés à Google on utilisait une clé publique *pub*
- Pour déchiffrer ces messages il fallait une **autre** clé *priv*, dite privée
- Tout le monde connaissait *pub* (e.g. installée dans les navigateurs)

Que pourrions nous faire ? (piste : on veut avoir une clé symétrique commune *sk*)
Envoyer une clé *sk* chiffrée avec *pub* puis on a un secret commun !

Plan

L'arrivée de la cryptographie à clé publique

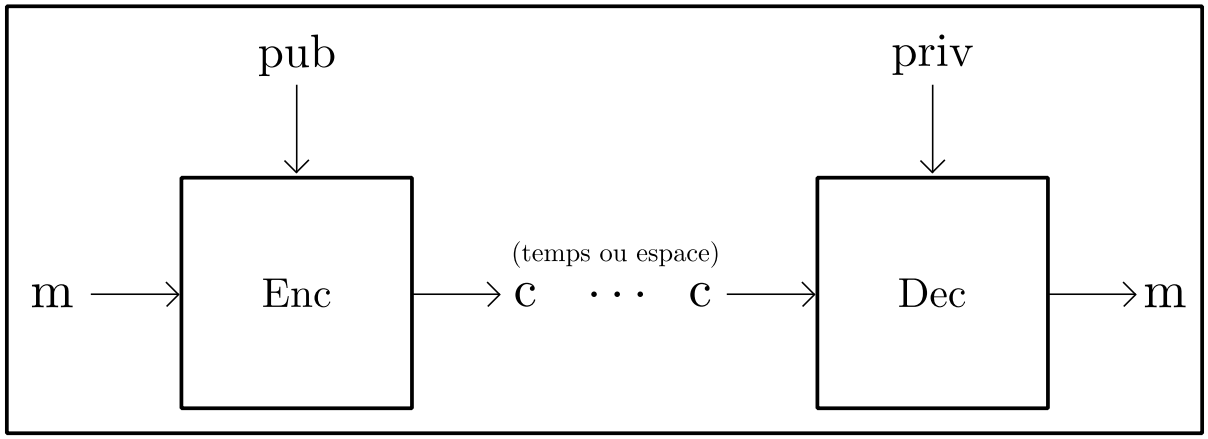


À quoi bon ?

Remplacer la confidentialité de la clé sk par l'intégrité de pub
Si mes interlocuteurs connaissent pub (publiée, hash vérifié, certifiée, etc.)
Ils peuvent **tous** m'envoyer des données chiffrées que seulement moi peut déchiffrer

Ils peuvent m'envoyer des clés (sk,ik) chiffrées puis après on fait du AES+HMAC

L'arrivée de la cryptographie à clé publique



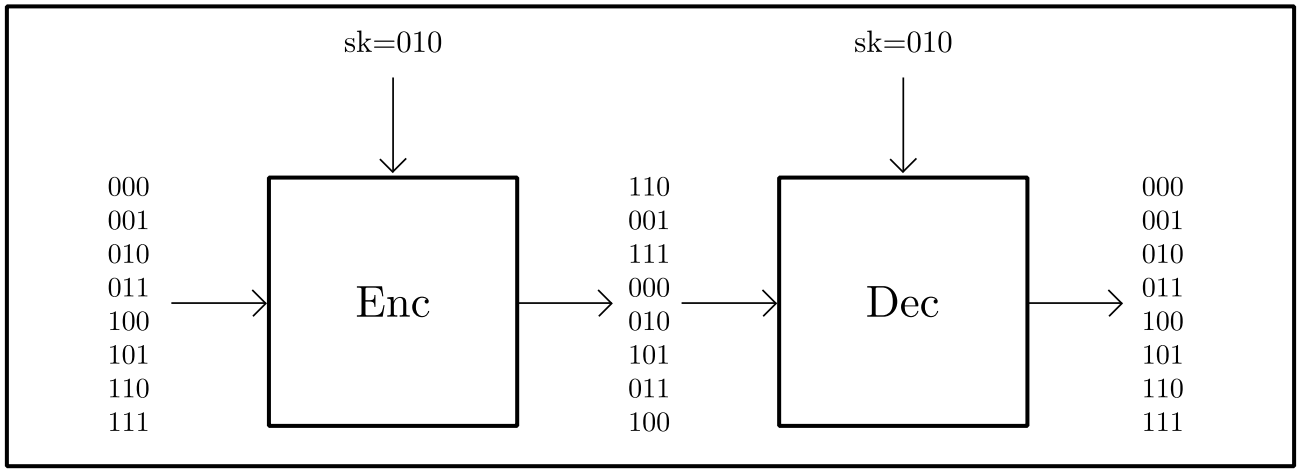
Une machine à laver ...

Inversible ET avec un design publique DONT on dévoilerait la clé ...
Comment ça peut être sûr ?

RSA (1977)

Rivest Shamir et Adleman
Premier protocole permettant de générer (*pub*, *priv*) tq dévoiler *pub* en gardant *priv* secret ne casse pas complètement la sécurité

Retour sur les chiffrements symétriques par blocs



Permutation

Quand on connaît sk :

- Enc permute les entiers de k bits
- Dec permute les entiers de k bits dans l'autre sens

On veut que (pub, Enc) fasse une permutation mais ne dévoile pas la permutation inverse si on ne connaît pas priv (permutation à trappe)

Permutations à trappe (1/2)

Exemple

Mettre au cube modulo 11 :

0	1	2	3	4	5	6	7	8	9	10
0	1	8	5	9	4	7	2	6	3	10

Permutation facile à calculer dans un sens. Et dans l'autre sens ?

Par exemple : racine cubique de 7 mod 17 ? (trouver m tq $m^3 \bmod 17 = 7$)

La magie du "lemme des bergers"

Peut on inverser $c = m^e \bmod N$?
Si e est premier alors, pour tout N, il existe un d tel que :

$$c = m^e \bmod N \Leftrightarrow m = c^d \bmod N$$

Si on connaît d on peut inverser la permutation facilement

Retour sur l'exemple

Inverse de $e = 3$ quand on travaille modulo 17 : c'est $d = 11$
Racine cubique de 7 modulo 17 : $7^{11} \mod 17 = 14$

Permutations à trappe (2/2)

Comment trouver d, inverse de e ?

On travaille modulo N :

- si N est premier c'est facile (par l'algorithme d'Euclide)
- si N est un produit de deux premiers $N = pq$
  - si on connait la factorisation facile (Euclide et CRT)
  - sinon **on ne sait pas faire**

Meilleur algo : factorisation 1024, 2048, 3072 bits $\rightarrow$ 80, 112, 128 bits de sécurité

Conclusion

Il suffit de prendre p, q suffisamment grands comme pour qu'on ne sache pas factoriser $N = pq \rightarrow$ permutation à trappe

Celui qui sait factoriser N peut appliquer la permutation dans un sens (x^e mod N) ou dans l'autre (x^d mod N) ... les autres ne savent qu'aller dans un sens.

Le cryptosystème RSA (Rivest, Shamir et Adleman)

GenKey$(k) = (pub, priv)$

Générer deux nombres premiers p, q de taille suffisante (attaques en 2^k)
Définir $N = pq$ et prendre $e = 65537$ $(= b1000000000000001)$
Calculer d l'inverse de e modulo $(p-1)(q-1)$
$(pub, priv) = ((N, e), d)$

Encrypt$(pub, m) = c$

$c = m^e \mod N$

Decrypt$(priv, c) = m$

$m = c^d \mod N$

Tailles et vitesses (1 coeur)

log(N)	Chiffrement	Déchiffrement	Sécurité (en bits)
1024	40K/s $\rightarrow$ 40Mbits/s	2K/s $\rightarrow$ 2Mbits/s	< 80
2048	11K/s $\rightarrow$ 22Mbits/s	320/s $\rightarrow$ 640Kbits/s	112

L'échange de clés

Idée

Avec RSA : envoi de matériel crypto chiffré avec une clé publique
Alternatives (au tableau) : Diffie-Hellman (DH), Elliptic-Curve DH (ECDH)

Usages

Protocoles d'échange de clés : $n^2 \rightarrow n$ clés échangées dynamiquement

Man-In-The-Middle

A envoie qqch à B, B envoie qqch à A ...
A envoie qqch, C l'intercepte et envoie qqch à B ...
Nécessité de mettre en place un système de certification
Très complexe dès qu'on a plus d'une organisation

Fin !

Des questions ?