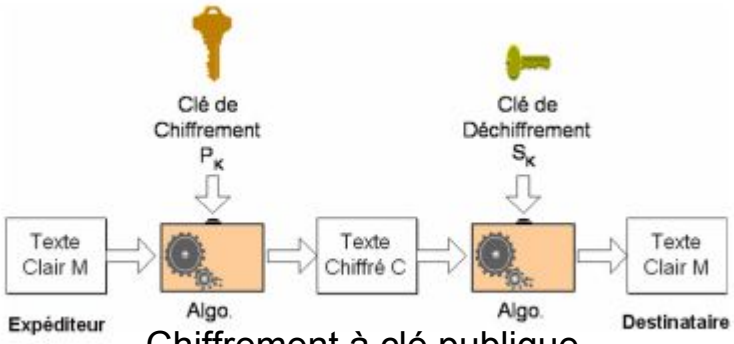
Les chiffrements par clé

Clé publique et clé privée

Le chiffrement par clé publique/asymétrique

Concept :

- Dans le cas des **systèmes asymétriques**, chaque personne possède 2 clés distinctes (une privée, une publique) . La clé **publique** sert à **crypter** le message. Le message ainsi codé est accessible à tous.
- Seul le **destinataire** qui possède une **clé privée** peut déchiffrer le message.



Chiffrement à clé publique

Le fonctionnement concret

La clé publique est visible par tout le monde dans un annuaire qui associe à chaque personne sa clé publique.

La clé privée n'est visible et connue que par son propriétaire. Il ne faut en aucun cas que quelqu'un d'autre que le propriétaire entre en possession de celle-ci.

Cas concret : exemple avec Bob et Alice

Comment Bob fait-il pour envoyer un message chiffré à Alice ?

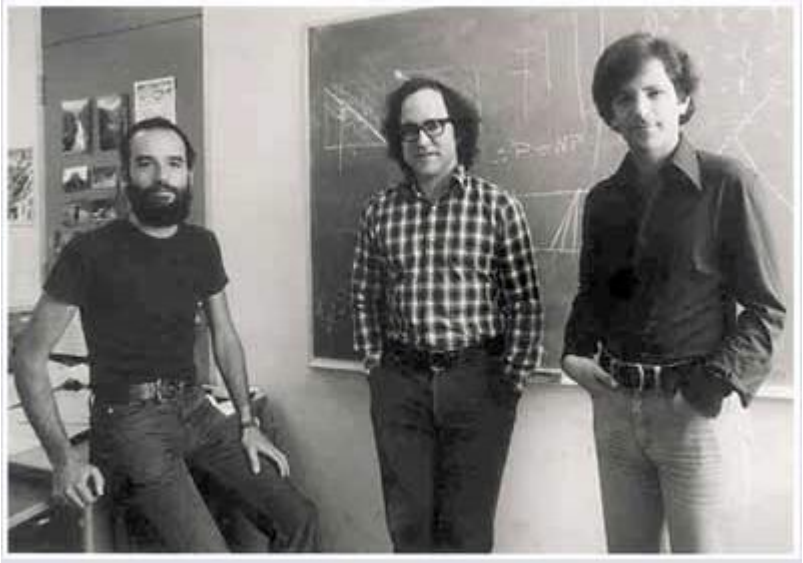
Bob va donc aller sur un annuaire et prendre la clé publique d'Alice.
Une fois qu'il la possède, il chiffre le message qu'il veut envoyer à Alice, puis le lui envoie.

Comment Alice déchiffre-t-elle le message reçu ?

Alice vient de recevoir le message chiffré de Bob, grâce à sa clé privée (qu'elle seule possède), elle peut déchiffrer et lire le message que Bob a envoyé

Un système asymétrique : le RSA

Il s’agit de l’algorithme à clé publique le plus connu, qui date de 1977. Il tient son nom de ses 3 inventeurs, Rivest, Shamir et Alderman.



Le cryptage des clés

La création des clés est basée sur le produit n de deux nombres premiers **p** et **q**.

Alice choisit deux **grands nombres premiers** p et q (+ de 100 chiffres) et calcule n=p*q. Elle choisit ensuite un nombre c (le coefficient de chiffrement)

La clé publique est constituée de **n** et **c. P** et **q** sont tenus secrets.

Exemple :

Alice choisit p=101 et q=113. n=p*q= 11 413. Elle choisit c=3 533. Alice transmet n et c sur un annuaire.

Le cryptage / décryptage

Bob récupère n et c. Il crypte son message m (un entier) et calcule m'=m^c, qu'il envoie à Alice. On calcule un nombre d par l'algorithme d'Euclide. On trouve d=6597

Alice reçoit m' et calcule m'^d mod n

Exemple :

n=11 413 et c=3 533. Bob veut envoyer un message (9 726). Il calcule m'=m^c, soit 9726^3 533. Il calcule mod 11 413 = 5 761 et envoie ce résultat.

Alice reçoit 5761 et calcule 5761^6597 mod 11 413 = 9726

La sécurité

Certains sites permettent de stocker et de publier les clés publiques des utilisateurs (annuaires électroniques). Ils se chargent aussi de les générer : on les appelle les PKI (Public Key Infrastructure) ou ICP (infrastructure à clé publique)

Sécurité de l’information ??

Un certificat associant identité de l’intervenant et clés peut être délivré.

La loi française autorise les clés de chiffrement jusqu'à 512 bits.

Avantages et inconvénients

Plus de problème pour communiquer la clé de déchiffrement -> les clés publiques peuvent être envoyées librement.

Le chiffrement par clés publiques = pas de code secret en commun

Il faut s'assurer que la clé publique récupérée est la bonne. Il n'y a pas d'identification mutuelle des intervenants.

Le système est aussi beaucoup plus long, il nécessite 1000 fois plus de temps de CPU

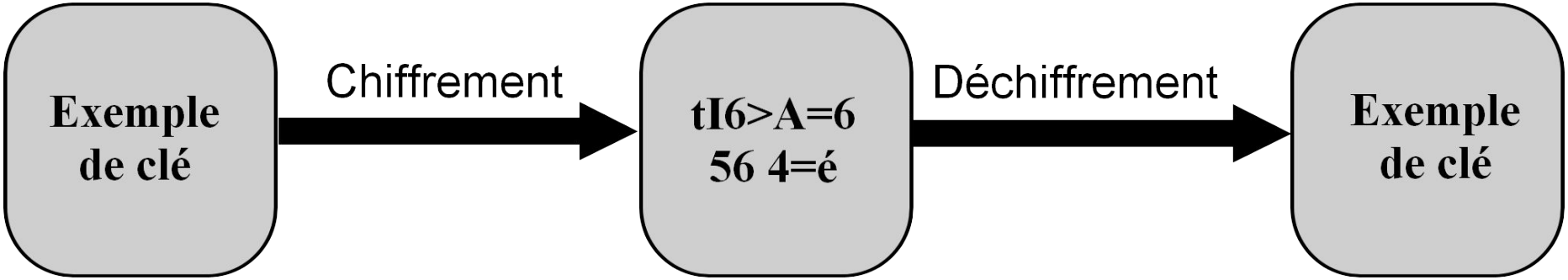
Le chiffrement par clé privée / symétrique

- chiffrement symétrique (permet de chiffrer et de déchiffrer des messages à l'aide d'une même clé)

- 2 catégories différentes :
- Par bloc : traité par blocs de données (entre 32 et 512 bits)
- Par flot : traité bit par bit

Exemple de chiffrement par clé privée / symétrique

Chiffrement par bloc (par bloc de données)	Chiffrement par flot (bit par bit)
- DES (Data Encrytion Standard) - AES (Advanced Encrytion Standard) - Feistel	- A5/1 - RC4 - Py - E0



Un chiffrement par flot : RC4

- conçu en 1987 par Ronald Rivest
- détails tenus secrets jusqu'en 1994 → posté de manière anonyme
- utilisé dans des protocoles tel que WPA (Wi-fi Protected Access)
- générateur de bits pseudo-aléatoire
  - permutation de 256 octets
  - deux pointeurs i et j qui servent d'index dans un tableau

Permutation

```
pour i de 0 à 255
    S[i] := i
finpour
j := 0
pour i de 0 à 255
    j := (j + S[i] + clé[i mod longueur_clé])
mod 256
    échanger(S[i], S[j])
finpour
```

- La permutation **S** se présente sous la forme d'un tableau de 256 entrées.

ex : [0] [1] [2] [3] [4] ... [255]

- L'algorithme de key schedule effectue 256 itérations

ex : [184] [106] [163] [64] [70] ... [201]

Flot pseudo-aléatoire

```
i := 0
j := 0
tant_que générer une sortie:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    échanger(S[i], S[j])
    octet_chiffrement = S[(S[i] + S[j]) mod 256]
    result_chiffré = octet_chiffrement XOR
octet_message
fintant_que
```

- Cet algorithme garantit que chaque valeur de **S** est échangée au moins une fois toutes les 256 itérations.

Avantages et inconvénients

- très rapides (en matériel et en logiciel)
- adaptés aux applications temps réel

- sécurité délicate

Sitographie

Chiffrement par clé privée :

- http://www.di.ens.fr/~bresson/P12-M1/P12-M1-Crypto_8.pdf
- http://iml.univ-mrs.fr/~rodier/Cours/Blocs.pdf
- https://fr.wikipedia.org/wiki/RC4