# La sécurité dans les réseaux

- Concepts de base
- Cryptographie traditionnelle
  - chiffrement par substitution
  - chiffrement par transposition
- Deux principes fondamentaux de la cryptographie
- Algorithmes à clés secrètes
  - algorithme DES
  - algorithme IDEA
- Algorithmes à clé publique
  - algorithme RSA
  - autres algorithmes à clé publique
- Non répudiation
- Protocoles d'authentification
- Signatures numériques

# Les concepts de base

- La confidentialité: assure la protection des données contre des attaques non autorisées.
- L'authentification: permet de s'assurer que celui qui se connecte est bien celui qui correspond au nom indiqué.
- L'intégrité: assure que les données reçues sont exactement celles qui ont été émises par l'émetteur autorisé.
- La non-répudiation: permet d'assurer qu'un message a bien été envoyé par une source spécifiée et reçu par un récepteur spécifié.
- Le contrôle d'accès: a pour but de prévenir l'accès à des ressources sous des conditions bien spécifiées et par des utilisateurs bien spécifiées.

# Deux principes fondamentaux de la cryptographie

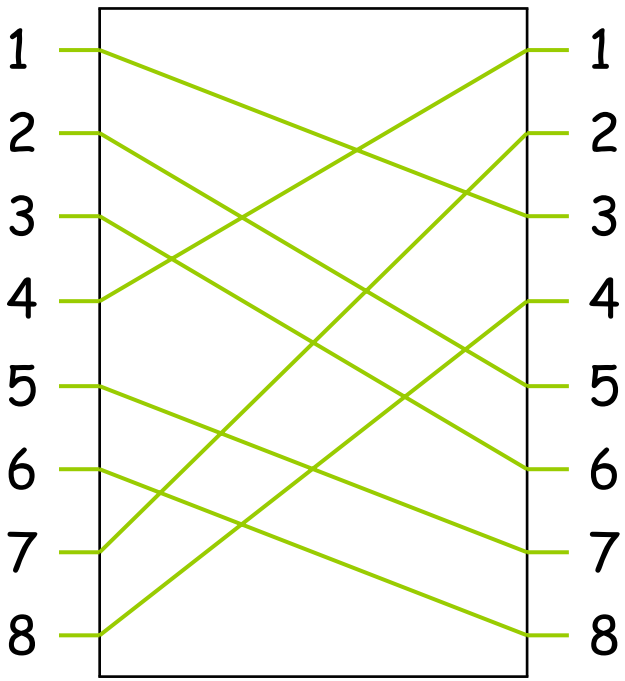
- Redondance: permet d'éviter qu'un intrus actif envoie de faux messages.
  - Facilite le décryptage
  - Ajout de la redondance via e.g. une suite aléatoire de mots français
- Eviter que des intrus actifs rejouent d'anciens scénarios.
  - Technique d'horodatage

# Algorithmes à clés secrètes

- Cryptographie traditionnelle: algorithmes relativement simples, clés plutôt longues.
- Cryptographie aujourd'hui: algorithmes sophistiqués et complexes, clés assez courtes.

# Algorithmes à clés secrètes

- Boîte-P (P-Box, P=Permutation). Permutation sur une entrée de 8 bits.



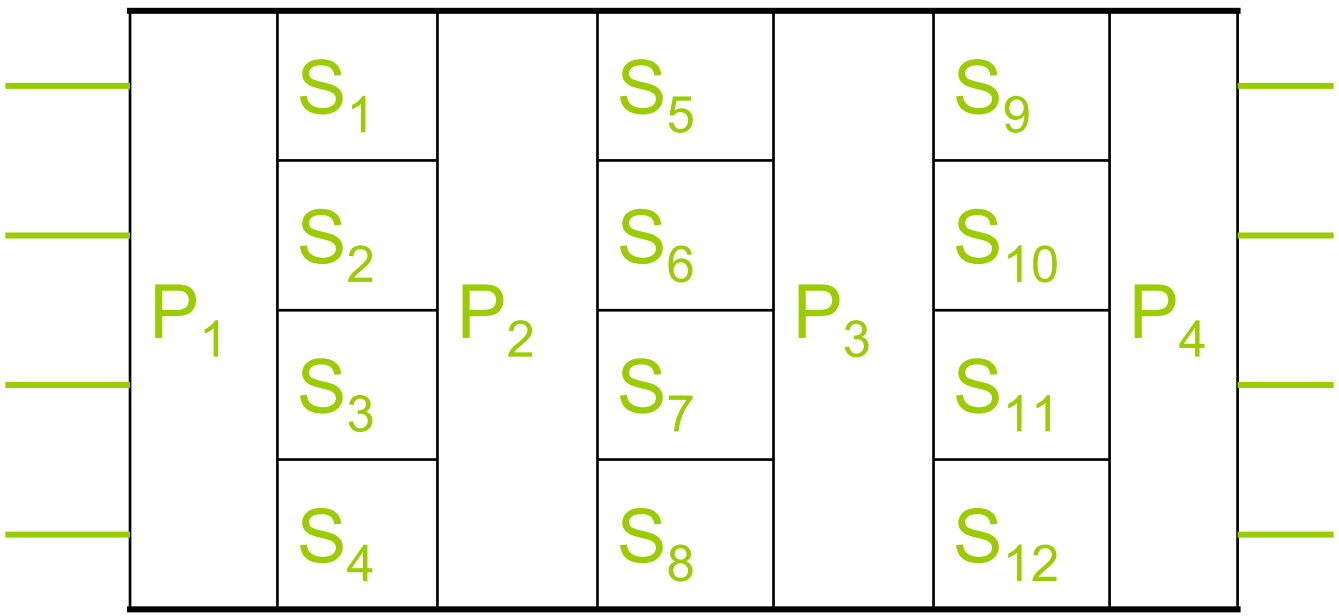

nombre de sorties
$= \leq \geq$
nombre d'entrées

# Algorithmes à clés secrètes

- Boîtes-S (S-box, S = substitution).
  - Étage 1: *p* bits d'entrée choisissent 1 des lignes de sortie du 1er étage
  - Étage 2: boîte-P
  - Étage 3: code en binaire sur *p* bits

# Algorithmes à clés secrètes

- Cascade de boîtes-S avec des boîtes-P



# Algorithme DES
## Data Encryption Standard

- Janvier 1977: gouvernement américain adopte la technique DES développée par IBM pour toutes les informations courantes non classifiées.
- Très vite adoptée par la suite par l'industrie
- Aujourd'hui: n'est plus sûre, mais encore très largement utilisée.

# Principe général du DES

- Texte en clair est codé par blocs de 64 bits
- Algorithme DES: paramétré avec une clé de 56 bits, comporte 19 étapes distinctes.
  - Étape 1: transposition sur les 64 bits du texte en clair, indépendante de la clé.
  - Étape 2 à 17: toutes identiques, mais paramétrées par différentes fonctions de la clé.
  - Étape 18: intervertit les 32 bits de droite et les 32 bits de gauche
  - Étape 19: transposition inverse de celle de l'étape 1

# Détail d'une itération de l'algorithme DES Encryptage

- $L_x = R_{x-1}$
- $R_x = L_{x-1} \oplus f_n (R_{x-1}, K_x)$
- $K_x$ sous clé de 48 bits
- $F_n$ bitwise fonction, appelée Feistel cipher
  - Construire un nombre E à partir de $R_{x-1}$, conformément à des règles de copie et de transposition fixées
  - Calculer un OU EXCLUSIF entre E et $K_x$
  - Découper le résultat en huit groupes de 6 bits, et charger chacun de ces groupes dans une boîte-S, chacune avec une clé différente. La boîte-P interne effectue une opération de compression en transposant les 64 bits de sortie à partir d'un décodeur 6 vers 64 sur 16 bits. Les 16 bits résultant sont alors passés dans un encodeur 16 vers 4 pour générer une valeur sur 4 bits.
  - Résultat final est la suite des huit sorties de 4 bits que l'on passe enfin dans une boîte-P.

# Détail d'une itération de l'algorithme DES

## Décryptage: mêmes étapes que pour l'encryptage

- $R_{x-1} = L_x$
- $L_{x-1} = R_x \oplus f_n (L_x, K_x)$
- $K_x$ sous clé de 48 bits
- $F_n$ bitwise fonction, appelée Feistel cipher

# Algorithm IDEA
## International Data Encryption Algorithm
## Lai and Massay, 1990

- Utilise une clé de 128 bits : résiste à toute recherche exhaustive de clé, loterie chinoise ou autres attaques par collision
- Seulement 8 itérations: chaque bit de sortie dépend de tous les bits de sortie

# Algorithmes à clé publique

- En 1976, Diffie et Hellman proposent une nouvelle façon de voir les choses. Plus de système tels que les clés de chiffrement et de déchiffrement sont les mêmes, mais
  - un algorithme de chiffrement (C) avec une clé dite publique
  - un algorithme de déchiffrement (D) avec une autre clé dite privée

  qui satisfont trois conditions
  - D(C(M)) = M
  - Il est excessivement difficile de déduire D à partir de C
  - C ne peut être cassé par une attaque du type texte en clair choisi.
- Utilisateur A: algorithme de chiffrement et la clé publique $C_A$ sont rendus publics.

# Communications entre deux correspondants qui n'ont jamais rien échangé

- Clés de chiffrement de $C_A$ et $C_B$ sont dans un annuaire public.
- A souhaite envoyer un message M à B:
  - Prend la clé pubique $C_B$ et calcule $C_B(M)$ qu'il transmet à B
  - B déchiffre le message à l'aide de sa clé privée: $D_B(C_B(M)) = M$
- Cryptographie asymétrique

# Algorithme RSA
## Rivest, Shamir et Adelman 1978

- Fondée sur certains principes de la théorie des nombres
- Algorithme
  - Choisir deux nombres premiers, p et q, chacun plus grand que $10^{100}$.
  - Calculer n = p.q et z = (p-1)(q-1).
  - Choisir un nombre d premier avec z
  - Chercher e tel que e.d = 1 (mod z)

# Opération de chiffrement avec l'algorithme RSA

- Chiffrer un message M
- Calculer $C = M^e \pmod n$
- Déchiffrer C: calculer $M = C^d \pmod n$.
- Pour tout $M \in [0,n[$, les deux fonctions de chiffrement et déchiffrement sont inverses l'une de l'autre
- Clé publique (e,n)
- Cle privé (d,n)

# Sécurité de l'algorithme RSA

- Difficulté de décomposer de très grands nombres en facteurs premiers. Si le crypanalyste pouvait factoriser n, il trouverait alors p et q, et donc z. La connaissance de z et e lui permettraient de trouver d en utilisant l'algorithme d'Euclide.
- Décomposition d'un nombre de 200 (500) chiffres : 4 milliards ($10^{25}$) d'années de calculs sur un ordinateur.

# Exemple

- p = 3, q =11 $\Rightarrow$ n =33 et z =20
- Choisissons d = 7 (7 et 20 n'ont pas de facteurs communs)
- Trouver e tel que 7e = 1 (mod 20)

  $\Rightarrow$ e = 3
- Chiffrement de C = suzanne

# Exemple (suite)

| Texte en clair (P) | | | Texte chiffré (C) | | Après déchiffrement | |
|---|---|---|---|---|---|---|
| Caractère | Valeur | $P^3$ | $P^3$ (mod 33) | $C^7$ | $C^7$ (mod 33) | Caractère |
| S | 19 | 6859 | 28 | 13492928512 | 19 | S |
| U | 21 | 9261 | 21 | 1801088541 | 21 | U |
| Z | 26 | 17576 | 20 | 1280000000 | 26 | Z |
| A | 01 | 1 | 1 | 1 | 1 | A |
| N | 14 | 2744 | 5 | 78125 | 14 | N |
| N | 14 | 2744 | 5 | 78125 | 14 | N |
| E | 05 | 125 | 26 | 8031810176 | 5 | E |

Calculs de l'émetteur

Calculs du récepteur

# Un autre exemple

- Pour créer la clé publique $K_p$:
  - Choisir deux grands nombres entiers positifs p et q ☞ **p = 7, q=17**
  - Calculer z = (p-1)(q-1) ☞ **z= 96**
  - Choisir un entier e qui soit un nombre premier avec z, et tel que e.d=1 (mod z) ☞ **e = 5**
  - Calculer n=p.q ☞ **n =119**
  - $K_p$ est la concaténation de n et e ☞ **$K_p$ = 119,5**

# Un autre exemple (suite)

- Pour créer la clé secrète $K_s$ :
  - Calculer d tel que mod(d×e, z) = 1
    ☞ **D × 5/96 = 1, d =77**
  - $K_s$ est la concaténation de n et d ☞ **$K_s$ = 119,77**

- Pour créer le texte chiffré C à partir de P:
  - Considérer P comme une valeur numérique ☞ **P=19**
  - $C = mod(P^e, n)$ ☞ **$C = mod(19^5, 119)$, C=66**

- Pour créer le texte chiffré P à partir de C:
  - $P = mod(C^d, n)$ ☞ **$P = mod(66^{77}, 119)$, P = 19**

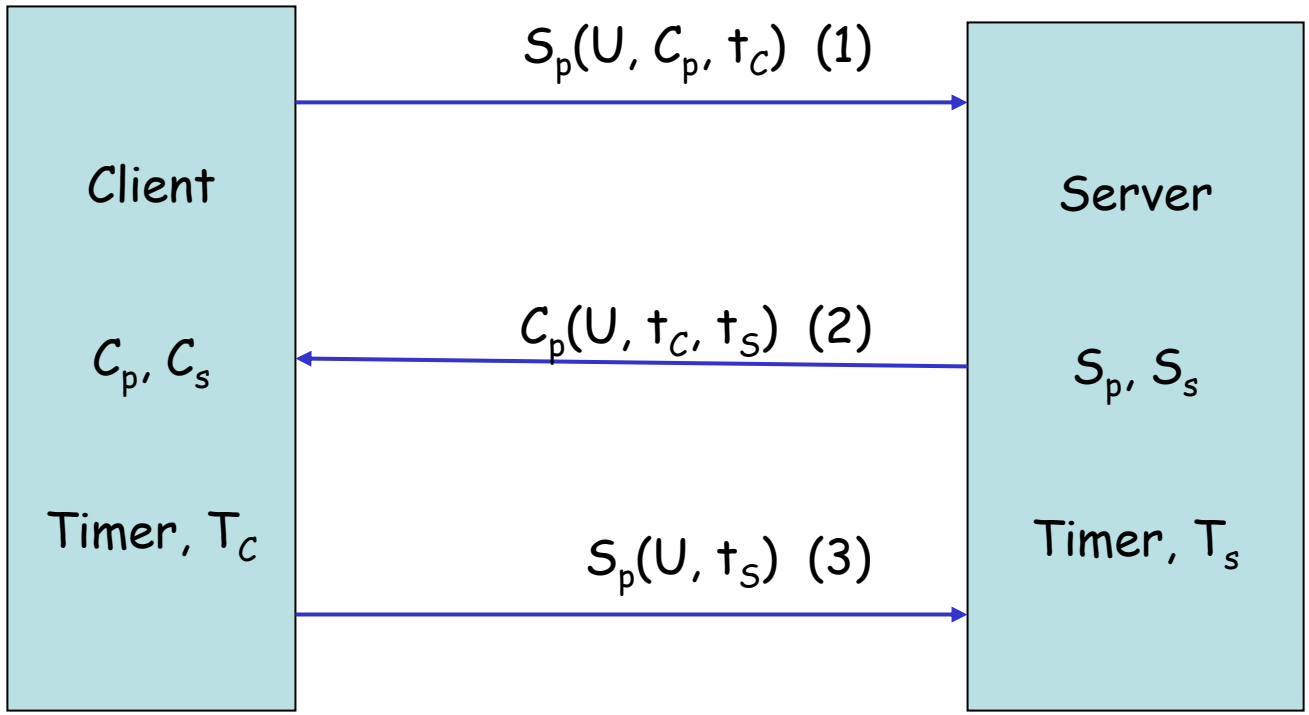
# Comment calculer $C = \mathrm{mod}(19^5, 119)$ ?

- $C = \mathrm{mod}(1 \times 19, 119) = 19$
- $C = \mathrm{mod}(19 \times 19, 119) = 4$
- $C = \mathrm{mod}(4 \times 19, 119) = 76$
- $C = \mathrm{mod}(76 \times 19, 119) = 16$
- $C = \mathrm{mod}(15 \times 19, 119) = 66$

$C = \mathrm{mod}(P^e, n) = ???$

$C \leftarrow 1$
Pour i variant de 1 à e faire
$C \leftarrow \mathrm{mod}(C \times P, n)$

# Protocoles d'authentification Schéma avec clés publiques




$C_p$, $C_s$ = client public/secret key
U = client user name

$S_p$, $S_s$ = server public/secret key
$t_C$, $t_S$ = client/server time-stamp

# Protocoles d'authentification Schéma avec clés publiques

- Tous les utilisateurs potentiels connaissent la clé publique $S_p$ du serveur.
- Client crée un message contenant son nom U, sa clé publique $C_p$, et un time-stamp $t_C$.
- Message est chiffré en utilisant $S_p$ et envoyé au serveur.
- Serveur déchiffre avec sa clé privé, $S_s$, et vérifie si il existe un usager enregistré avec le nom U.
- Si oui, le serveur répond avec un message comprenant le nom de l'usager, U, un time stamp, $t_C$, et un autre time stamp pour indiquer quand la réponse a été faite par le serveur, $t_S$.
- Le serveur garde une copie de sa réponse et chiffre le message en utilisant la clé public du client, $C_p$. La réponse est envoyée au client (2).
- Sur réception de la réponse, le client déchiffre le message en utilisant sa clé privé, $C_s$, et en déterminant si le $t_C$ est le même que celui qu'il a envoyé, suppose qu'il a été authentifier correctement par le serveur.
- Client poursuit en accusant réception avec un second message contenant son nom U + serveur time-stamp $t_S$. Ceci est chiffré et envoyé au serveur en utilisant sa clé publique $S_p$. (3)
- Le serveur déchiffre en utilisant sa clé secrète $S_s$ et en déterminant si $t_s$ est le même que celui envoyé, se préparer à accepter des requêtes pour des services de la part du client.