**Année scolaire 2025 – 2026**

Lycée Victor Hugo - Colomiers

# Terminale NSI

---

# ARSE
# Architecture – Réseaux – Systèmes d'exploitation

---

**Thomas Ricart**
ricart.lvh@laposte.net

24 mars 2025

# ARSE01 – System On Chip

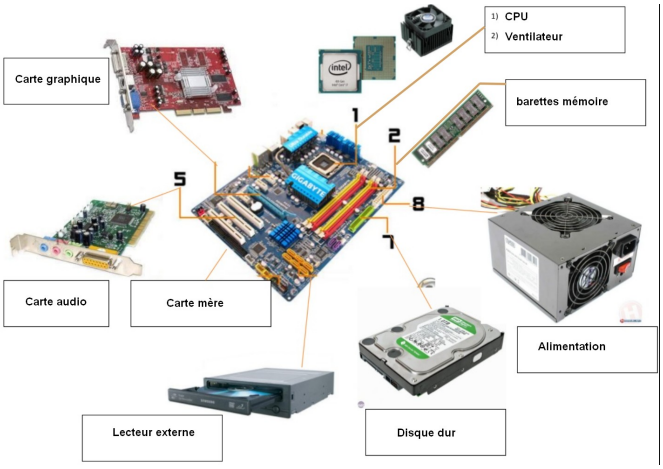
## 1. Architecture matérielle

Dans la suite du cours nous nommerons *ordinateur* l'objet général, qu'il soit PC fixe, PC portable ou Système sur puce.

**Définition 1: Composants d'un ordinateur**

Qu'ils soient intégrés ou indépendants, les différents composants d'un ordinateur peuvent être regroupés en plusieurs catégories:

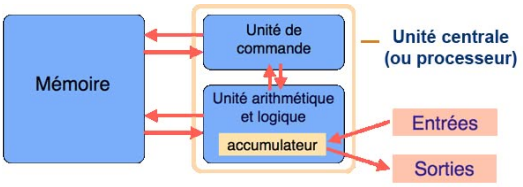
- les périphérique d'entrée et de sortie
- la carte mère
- la mémoire
- le processeur
- les cartes périphériques
- l'alimentation




**Définition 2: Architecture Von vonNeumann**

L'architecture de von Neumann décompose l'ordinateur en 4 parties distinctes:

- **l'unité arithmétique et logique (UAL)** ou unité de traitement: effectue les opérations de base.
- **l'unité de contrôle** chargé de gérer le séquençage des opérations;
- **la mémoire** qui contient à la fois les données et le programme qui indique à l'unité de controe quels sont les calculs à effectuer;
- **les dispositifs d'entrée / sortie** pour communiquer avec le monde extérieur.



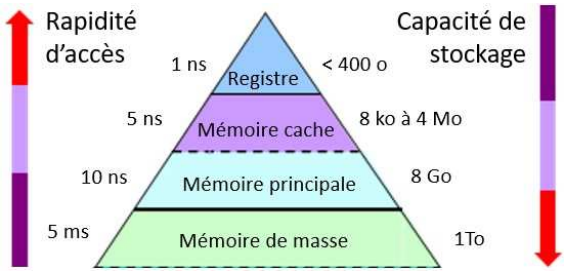

**Définition 3: La mémoire**

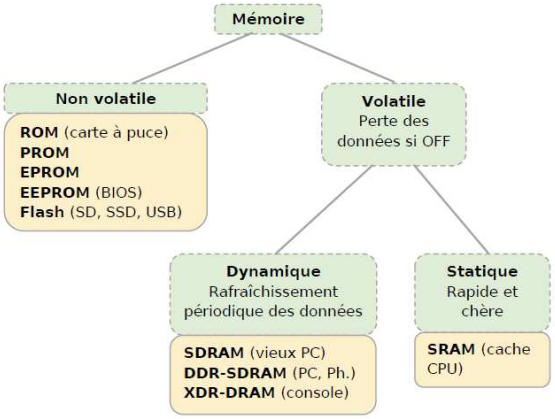
La mémoire utilise le magnétisme dans les disques durs (HDD pour Hard Drive Disc) ou l'électricité pour la RAM, les clés USB, les SSD (Solid-State Drive) pour stocker les bits d'information. Elle peut être :

- **permanente**, quand les données sont conservées même sans alimentation électrique,
- **volatile**, lorsqu'elle sont perdus sans alimentation électrique.

Dans un ordinateur, il existe plusieurs familles de mémoire que l'on peut hiérarchiser ainsi :

- Les **registres** sont des mémoires de taille réduite mais d'accès extrêmement rapide car directement intégrée dans le processeur. Ils stockent l'instruction, les opérandes et le résultat.
- La **mémoire cache** est une mémoire rapide destinée à accélérer l'accès à la mémoire centrale en stockant les données les plus utilisées.
- La mémoire principale ou **mémoire vive** ou **RAM** (Random Access Memory) est l'espace principal de stockage du processeur. C'est une mémoire volatile qui au démarrage d'un ordinateur stocke d'abord les instructions et les données du système d'exploitation puis des programmes à exécuter. Quand un programme est fermé, il est déchargé de la mémoire vive.
- La **mémoire de masse** (mémoire de stockage) est permanente et de grande capacité.




**Définition 4: Le processeur CPU**

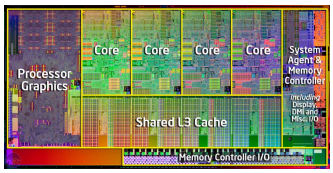
Un **microprocesseur** est un circuit intégré sur puce composé de milliards de transistors permettant d'exécuter des instructions. La taille des CPU est de plus en plus réduite grâce aux avancées scientifiques qui permettent des techniques de gravures en-dessous de la dizaine de nanomètres.

Historiquement, il existe deux familles de microprocesseurs :

- les processeurs RISC (Reduced Instruction Set Computer) qui disposent de peu d'instructions mais très rapide à effectuer,
- les processeurs CISC (Complex Instructions Set Computer) qui proposent des instructions plus élaborées mais qui nécessitent un temps plus long d'exécution.

Un processeur fonctionne de manière cadencées. Il est caractérisé par la fréquence à laquelle il réalise un cycle d'exécution. Pour les processeurs d'aujourd'hui cette fréquence est de l'ordre de quelques GigaHertz soit 1 milliards d'opérations par seconde.

Il peut être constitué de plusieurs cœurs, ce qui lui permet d'exécuter des instructions en parallèle, et d'augmenter encore ses performances. Cela entraine néanmoins une augmentation de la consommation et donc de la température.




## 2. Système sur Puce

**Définition 5: System On Cip (SOC)**

Le principe d'un système sur puce ou System On a Chip (SoC) est d'intégrer au sein d'une puce unique un ensemble de composants habituellement physiquement dissociés dans un ordinateur classique (ordinateur de bureau ou ordinateur portable). On peut retrouver ainsi au sein d'une même puce:

- le microprocesseur (CPU)
- la carte graphique (GPU)
- la mémoire RAM
- éventuellement des composants de communication (WiFi, Bluetooth...)

**Propriété 1: Avantages et Inconvénients**

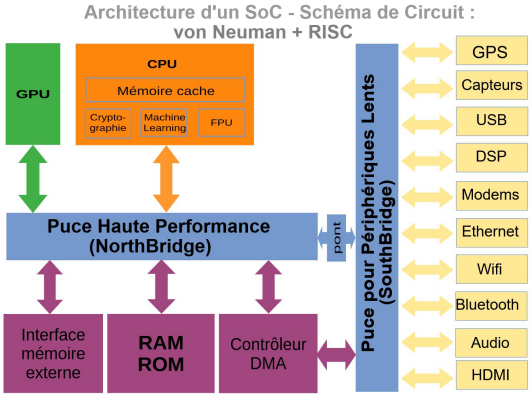
`Avantages`

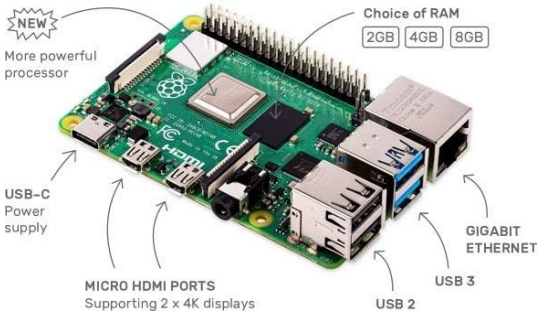
- moindre consommation électrique
- moindre encombrement
- pas besoin de refroidissement
- meilleure sécurité (vue globale sur la sécurité qui n'est plus dépendante d'une multitude de composants)
- moindre coût (forte automatisation du processus, gros volumes de production)

`Inconvénients`

- Impossibilité de choisir indépendamment ses composants
- Pas de mise à jour possible / remplacement / ajout d'un composant
- La panne d'un seul composant entraine la panne totale du SoC

## Propriété 2: Des exemples de SoC




**Exercice 1:** Choisir la bonne réponse

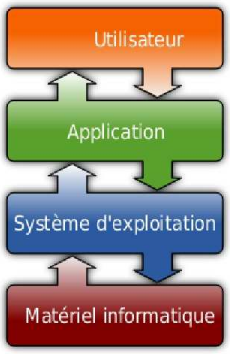
1. Dans le modèle d'architecture de Von Neumann, que signifie UAL?
   A) Unité Additive et logique
   B) Unité Active et logique
   C) Unité Active de Lecture
   D) Unité Arithmétique et Logique
2. Parmi tous les registres internes que possède une architecture mono-processeur, il en existe un appelé compteur `ordinal program counter`. Quel est le rôle de ce registre ?
   A) il contient le nombre d'opérandes utilisés
   B) il contient le nombre d'instructions contenues dans le programme
   C) il contient l'adresse mémoire de la prochaine instruction à exécuter
   D) il contient l'adresse mémoire de l'opérande à récupérer
3. Quel appareil ne possède généralement pas de SoC?
   A) un ordinateur
   B) une console de jeu
   C) un smartphone
   D) une tablette
4. Quel est le rôle de l'unité arithmétique et logique dans un ordinateur?
   A) effectuer les opérations
   B) stocker les opérations
   C) afficher les opérations
   D) définir l'ordre des opérations
5. Les avantages des SoC:
   A) faible consommation énergétique
   B) plus facile de remplacer un composant défectueux
   C) faible cout de production
   D) puissance plus élevée
6. Dans le modèle d'architecture de Von Neumann, quel est le rôle de l'unité de contrôle?
   A) gérer le séquençage des opérations
   B) effectuer les opérations
   C) stocker les données
   D) lire les données
7. Dans l'architecture générale de Von Neumann, la partie qui a pour rôle d'effectuer les opérations de base est
   A) la mémoire
   B) l'unité centrale
   C) les dispositifs d'entrée / sortie
   D) l'unité arithmétique et logique
8. Les données et les programmes sont stockées dans:
   A) l'unité logique et arithmétique
   B) la mémoire
   C) l'unité de contrôle
   D) les entrées / sorties

# ARSE02 – Les processus

## 1. Les systèmes d'exploitation (OS)

Souvent appelé l'OS (Operating System), le système d'exploitation est un ensemble de programme qui permet d'utiliser les éléments physiques d'un ordinateur pour exécuter les applications nécessaires à l'utilisateur. L'élément fondamental du système d'exploitation est son noyau, c'est lui qui permet l'accès aux ressources matérielles.

Toute machine est doté d'un OS qui a pour fonction:




- de charger les **programmes** depuis la mémoire de masse,
- de **gérer leur exécution** en leur créant des processus et en ordonnançant les tâches,
- de **gérer les ressources** (microprocesseur, mémoire, disques, carte graphique, carte réseau, clavier, souris, ...)
- de **gérer des accès aux ressources** pour permettre: d'une part à tous les utilisateurs de travailler simultanément, et d'autre part de ne permettre l'utilisation d'une ressource qu'aux utilisateurs autorisés.
- d'**assurer la sécurité globale du système**.

## 2. Les processus

**Définition 1: Les processus**

Un **processus** est une **instance** de programme en cours d'exécution sur un ordinateur. Il est caractérisé par:

- un ensemble d'instructions à exécuter souvent stockées dans un fichier sur lequel on clique pour lancer un programme (par exemple firefox.exe)
- un espace mémoire dédié à ce processus pour lui permettre de travailler sur des données et des ressources qui lui sont propres: si vous lancez deux instances de firefox, chacune travaillera indépendamment l'une de l'autre.
- des ressources matérielles: processeur, entrées-sorties (accès à internet en utilisant la connexion Wifi).

Un **processus** est un **programme** en cours d'exécution. Il ne faut donc pas confondre les deux. Il y a une différence entre programme et processus: le même programme peut être lancé plusieurs fois et donner naissances à des processus distincts.

**Méthode 1: Observation des processus**

Sur chaque OS, il y a des utilitaires en mode graphique (GUI) ou en mode console (CLI) pour observer les processus en cours d'exécution.

- Sous Windows: gestionnaire des taches (`CTRL+ALT+SUPP`).
- Sous Ubuntu: Moniteur systeme (`Menu/Outils Systeme`).

Sur les postes du lycée, l'accès au gestionnaire des tâches est verrouillé. Il est possible de créer un raccourci vers le terminal Windows Powershell avec un clic droit sur le bureau puis "nouveau raccourci" avec l'adresse : `C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe`. La commande `ps` permet de visualiser les processus comme dans le gestionnaire des tâches.

- Sous Linux, depuis le terminal, la commande pour afficher les processus est également `ps` (`-e`, `-f`) ou `top`. Utiliser l'émulateur du Terminal Linux en ligne JSLinux pour tester cette commande.

## 3. Etats des processus

L'OS doit permettre à toutes les applications et tous les utilisateurs de travailler en même temps, mais cette simultanéité n'est qu'illusion. En réalité, il va donner l'impression à chacun qu'il est seul à utiliser l'ordinateur et ses ressources physiques. Cela nécessite une gestion complexe des processus qui est réalisée par une partie spécifique du noyau du système d'exploitation, appelé : l'**ordonnanceur** (scheduler).

Un cœur de processeur ou un périphérique ne peuvent pas être partagés. Ce qui est réellement partagé c'est leur temps d'utilisation. L'ordonnanceur alloue la ressource à tour de rôle à chaque processus sur un intervalle de temps très courts qui peut varié selon les besoin. Le plus petit intervalle de temps allouable est appelé le **quantum de temps** (20 à 50 ms suivant les OS et les algorithmes d'ordonnancement utilisés).

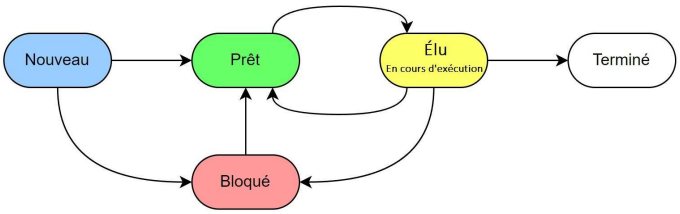
Les objectifs de l'ordonnanceur sont :

- de minimiser le temps de traitement du processus d'un utilisateur,
- de garantir l'équité entre les différents processus,
- d'optimiser l'utilisation des ressources,
- d'éviter les blocages.

**Définition 2: Etats des processus**

C'est dans un registre (mémoire) que les instructions d'un processus en cours d'exécution sont copiées. Lorsqu'un processus est **interrompu** pour laisser la place à un autre processus, l'état des registres au moment de l'interruption doit être sauvegardé afin que lorsque ce sera de nouveau son tour d'être exécuté, son **contexte d'exécution** soit parfaitement rétabli et qu'il puisse poursuivre son exécution.

Tous les processus qui cohabitent en mémoire à un instant donné ne sont pas forcément en attente d'un quantum de temps processeur. Ils peuvent **attendre** d'avoir un accès à une autre **ressource** (périphérique d'entrée/sortie) ou bien ils peuvent également attendre le résultat d'un autre processus. Dans ces cas là, on dit qu'il est dans l'état "**bloqué**". Au contraire quand un processus a tout ce qui lui faut et n'attend qu'un quantum de temps processeur, on dit qu'il est dans l'état "**prêt**".




**Exercice 1:** Associer chacune des descriptions suivantes à l'état du processus correspondant:

1. Un processus a été attaché à une instance d'un programme et il n'a encore jamais été exécuté. Son contexte d'exécution initial doit être préparé. .....
2. La dernière instruction de processus a été exécutée. Ses ressources affectées et son environnement mémoire vont être libérés.
3. Un ou plusieurs quantums de temps processeur ont été alloués au processus. Son contexte d'exécution a été copié dans les registres et ses instructions sont exécutées jusqu'à l'écoulement du temps imparti. .....
4. Le processus est en attente d'une ou plusieurs ressources (réseau, périphériques d'entrée/sortie, action de l'utilisateur, ...) ou du résultat d'un autre processus. Il n'est pas éligible à un quantum de temps processeur. .....
5. Le contexte d'exécution est sauvegardé et complet. Le processus est en attente de quantums de temps processeur.

# 4. Ordonnancement des processus

**Définition 3: Ordonnancement préemptif et non préemptif**

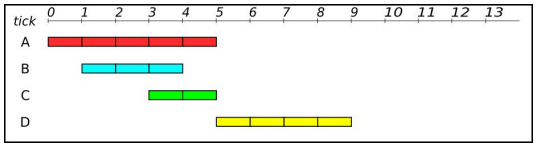
- un **ordonnancement non préemptif** est un ordonnancement où tout processus démarré n'a aucune raison de s'arrêter avant la fin. Si le système n'est pas préemptif, on peut penser à une organisation de type **file** (Premier arrivé, premier sorti: FIFO). (exemple: file d'attente d'impression).
- un **ordonnancement préemptif** est un ordonnancement où la priorité et donc l'élection peut être donnée à n'importe quel processus suivant un critère prédéfini. Si le système est préemptif, on réévalue le critère de priorité à chaque tour d'horloge. (exemple: prioriser les processus les plus courts, les plus longs, les plus rapides à se terminer, les plus *prioritaires*...)

**Définition 4: Quelques définitions**

- Le **temps de séjour** d'un processus est la différence entre le temps de terminaison et le temps d'arrivée;
- Le **temps d'attente**, est le temps de séjour auquel on retranche le temps d'exécution.

**Méthode 2: Dans la suite des exemples, nous prendrons les 4 processus suivants:**

Imaginons que le processeur ait à exécuter 4 processus, dont les temps d'exécutions sont différents, et qui se sont présentés à différents instants au processeur :



## 4.1. Les systèmes non préemptifs

**Méthode 3: Méthode FCFS: Firt Come, First Serve**

**FCFS**: Fist Come, First Serve. Les processus sont stockés dans une file dans l'ordre d'arrivée des processus.

1. Proposer le graphe d'exécution de cette méthode

2. Calculer le temps de séjour nécessaire à l'exécution de chaque processus .....
3. Calculer le temps de séjour moyen .....
4. Calculer le temps d'attente de chaque processus .....
5. Calculer le temps d'attente moyen .....

**Méthode 4: Méthode SJF: Short Job First**

**SJF**: Shortest Job First. C'est difficile d'évaluer le temps d'exécution d'un processus mais c'est ce critère qui est utilisé pour prioriser leurs exécutions. C'est typiquement cet algorithme qui est appliqué aux caisses d'un supermarché, lorsqu'un client laisse passé un autre client ayant peu d'article.

1. Proposer le graphe d'exécution de cette méthode

2. Calculer le temps de séjour nécessaire à l'exécution de chaque processus .....
3. Calculer le temps de séjour moyen .....
4. Calculer le temps d'attente de chaque processus .....
5. Calculer le temps d'attente moyen .....

## 4.2. Les système préemptifs

**Méthode 5: Méthode SRT: Shortest Remaining Time**

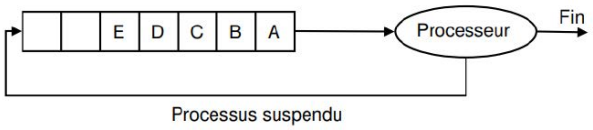
**SRT**: Shortest Remaining Time. C'est difficile d'évaluer le temps d'exécution d'un processus mais c'est ce critère qui est utilisé pour prioriser leurs exécutions.
L'ordonnancement du plus petit temps de séjour ou Shortest Remaining Time (SRT) est la version préemptive de l'algorithme SJF. Un processus arrive dans la file de processus, l'ordonnanceur compare la valeur espérée pour ce processus contre la valeur du processus actuellement en exécution. Si le temps du nouveau processus est plus petit, il rentre en exécution immédiatement.

1. Proposer le graphe d'exécution de cette méthode

2. Calculer le temps de séjour nécessaire à l'exécution de chaque processus .....
3. Calculer le temps de séjour moyen .....
4. Calculer le temps d'attente de chaque processus .....
5. Calculer le temps d'attente moyen .....

**Méthode 6: Méthode Round - Robin**

L'**algorithme du tourniquet ou Round Robin** (RR) représenté sur la figure ci-dessous est un algorithme ancien, simple, fiable et très utilisé. Il mémorise dans une file du type FIFO (First In First Out) la liste des processus prêts, c'est-à-dire en attente d'exécution.




1. si un nouveau processus est créé, il est mis dans la file d'attente;
2. ensuite on défile un processus de la file d'attente et on l'exécute pendant un quantum de temps;
3. si le processus exécuté n'est pas terminé, on le replace dans la file.

1. Proposer le graphe d'exécution de cette méthode pour un quantum de 1

2. Calculer le temps de séjour nécessaire à l'exécution de chaque processus .....
3. Calculer le temps de séjour moyen .....
4. Calculer le temps d'attente de chaque processus .....
5. Calculer le temps d'attente moyen .....

## 5. Interblocages

Les interblocages sont des situations de la vie quotidienne. Un exemple est celui du carrefour avec priorité à droite où chaque véhicule est bloqué car il doit laisser le passage au véhicule à sa droite.

**Définition 5: Interblocages**

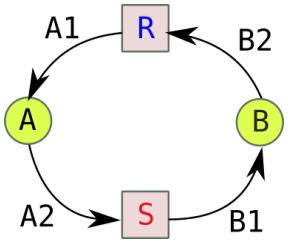
En informatique également, l'interblocage peut se produire lorsque des processus concurrents s'attendent mutuellement. Les processus bloqués dans cet état le sont définitivement. Ce scénario catastrophe peut se produire dans un environnement où des ressources sont partagées entre plusieurs processus et l'un d'entre eux détient indéfiniment une ressource nécessaire pour l'autre.

**Exemple 1: Exemple d'interblocage**

Considérons 2 processus A et B, et deux ressources R et S. L'action des processus A et B est décrite ci-dessous:

- A demande R et l'obtient
- B demande S et l'obtient
- A demande S bloqué par B. Il passe son tour
- B demande R bloqué par A. Il passe son tour
- ...

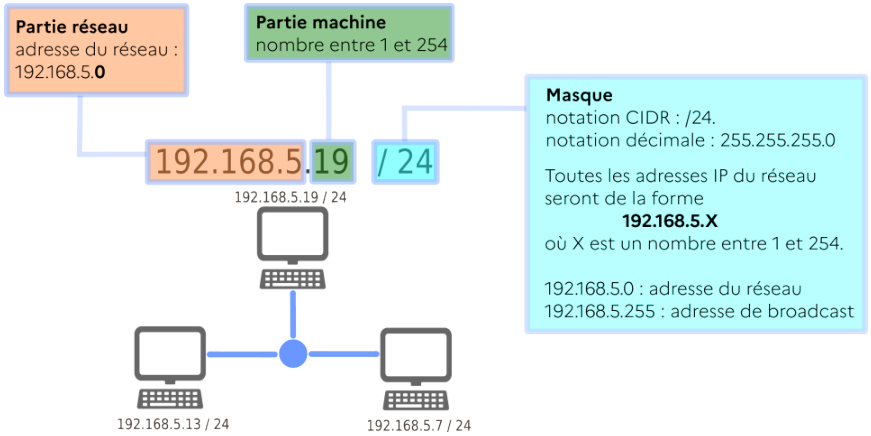
| Processus A | | Processus B | |
|---|---|---|---|
| étape A1 : | demande R | étape B1 : | demande S |
| étape A2 : | demande S | étape B2 : | demande R |
| étape A3 : | libère S | étape B3 : | libère R |
| étape A4 : | libère R | étape B4 : | libère S |




Les deux processus A et B sont donc dans l'état Bloqué, chacun en attente de la libération d'une ressource bloquée par l'autre: ils se bloquent mutuellement. Cette situation (critique) est appelée **interblocage** ou **deadlock**.

# ARSE03 – Protocoles de routage

## 1. Les notions de réseaux de la classe de Première

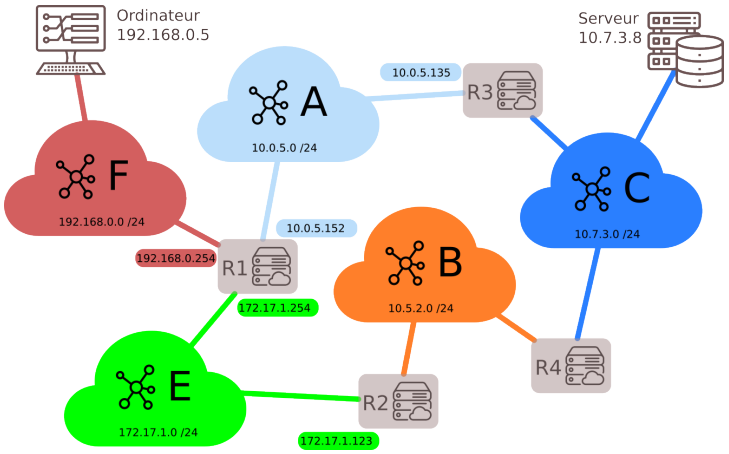



Lorsqu'une machine A, d'adresse `IP_A` veut discuter avec une machine B, d'adresse `IP_B`:

- La machine A calcule (grâce au masque de sous-réseau) si B est dans le même sous-réseau qu'elle, ou pas.
- Si oui, elle peut donc connaitre l'adresse MAC de la carte réseau de la machine B (soit elle la possède déjà dans sa table ARP, soit elle la demande en envoyant un message de `broadcast` à tout le sous-réseau: "qui possède cette adresse `IP_B`?"). Elle envoie donc dans le sous-réseau une trame ayant pour entête l'adresse MAC de B: le switch lit cette trame, sait sur quel port est branché la machine B et lui envoie spécifiquement donc le message.
- Si B n'est pas dans le même sous-réseau que A, A mettra en entête de sa trame l'adresse MAC de la carte réseau du routeur, qui joue le rôle de passerelle. Le routeur va ouvrir la trame et va observer l'`IP_B`, à qui il doit remettre ce message. C'est maintenant que vont intervenir les protocoles de routage:
  - est-ce que B est dans le même sous-réseau que le routeur?
  - est-ce que B est dans un autre sous-réseau connu du routeur?
  - est-ce que B est totalement inconnu du routeur?

Ces questions trouveront des réponses grâce à table de routage du routeur.

## 2. Tables de routage




**Définition 1: Le routeur**

Dans notre exemple, R1, R2, R3 et R4 sont des routeurs. Ils servent de **passerelle** entre deux sous – réseaux. Chacune de ses connexions avec un réseau est appelée une **interface** repérée par une adresse IP qui lui est attribuée par ce réseau. Par exemple R1 possède 3 interfaces:

- `10.0.5.152` dans le réseau `10.0.5.0/24`
- `192.168.0.254` dans le réseau `192.168.0.0/24`
- `172.17.1.254` dans le réseau `172.17.1.0/24`

Les tables de routage sont des informations stockées dans le routeur permettant d'aiguiller intelligemment les données qui lui sont transmises. Dans le réseau ci-dessus, si l'ordinateur d'adresse `192.168.0.5` veut interroger le serveur `10.7.3.8`:

- l'adresse `10.7.3.8` n'étant pas dans le sous-réseau F (d'adresse `192.168.0.0/24`), la requête est confiée au routeur R1 via son adresse passerelle dans le réseau F (ici `192.168.0.254`).
- le routeur R1 observe si l'IP recherchée appartient à un autre des sous-réseaux auquel il est connecté. Ici, l'IP recherchée `10.7.3.8` n'appartient ni au sous-réseau A, ni au sous-réseau E.
- le routeur R1 va donc regarder dans sa table de routage l'adresse passerelle d'un autre routeur vers qui elle doit rediriger les données. Si le sous-réseau C fait partie de sa table de routage, le routeur R1 saura alors que le meilleur chemin est (par exemple) de confier les données au routeur R3.
- si le sous-réseau C ne fait pas partie de la table de routage, le routeur R1 va alors le rediriger vers une route par défaut (que l'on peut assimiler au panneau toutes directions sur les panneaux de signalisation).

**Définition 2: Interface et Passerelle**

Les tables de routage des routeurs font très souvent apparaitre deux colonnes, interface et passerelle, dont il ne faut pas confondre l'utilité:

- **interface**: c'est l'adresse IP de la carte réseau du routeur par où va sortir le paquet à envoyer. Il y a donc toujours une adresse d'interface à renseigner (car un paquet sort bien de quelque part !). Parfois cette interface sera juste nommée `interface1` ou `interface2`.
- **passerelle**: c'est l'adresse IP de la carte réseau du routeur à qui on va confier le paquet, si on n'est pas capable de le délivrer directement (donc si l'adresse IP de destination n'est pas dans notre propre sous-réseau). Cette adresse de passerelle n'est donc pas systématiquement mentionnée. Quand elle l'est, elle donne le renseignement sur le prochain routeur à qui le paquet est confié.

**Exercice 1:** Compléter la table de routage du routeur R1

- Les trois réseaux F, A et E sont directement accessibles au routeur R1, puisqu'il en fait partie: il n'a donc pas besoin d'adresse passerelle pour communiquer avec ces réseaux.
- Par contre, la communication avec le réseau B nécessite de confier le paquet au routeur R2 (c'est le choix de cette table de routage). Il faut donc mentionner l'adresse IP de ce routeur R2 (`172.17.1.123`), qu'on appelle adresse de passerelle.
- De la même manière, la communication avec le réseau C nécessite de confier le paquet au routeur R3 (c'est le choix de cette table de routage). Il faut donc mentionner l'adresse IP de ce routeur R3 (`10.0.5.135`).

| Dest. | Interface | Passerelle |
|---|---|---|
| A | | |
| B | | |
| C | | |
| E | | |
| F | | |

**Définition 3: Comment sont construites ces tables de routage?**

- Soit à la main par l'administrateur réseau, quand le réseau est petit: on parle alors de table statique.
- Soit de manière dynamique: les réseaux s'envoient eux-mêmes des informations permettant de mettre à jour leurs tables de routages respectives. Des algorithmes de détermination de meilleur chemin sont alors utilisés.

# 3. Protocole RIP

**Définition 4: Protocole RIP**

Le Routing Information Protocol (RIP) est basé sur l'échange (toutes les 30 secondes) des tables de routage de chaque routeur. Au début, chaque routeur ne connait que les réseaux auquel il est directement connecté (distance 1).

Ensuite, chaque routeur va recevoir périodiquement la table des réseaux auquel il est connecté, et mettre à jour sa propre table suivant les règles ci-dessous:

- s'il découvre une route vers un nouveau réseau inconnu, il l'ajoute à sa table en augmentant la distance annoncée de 1.
- s'il découvre une route vers un réseau connu mais plus courte (en rajoutant 1) que celle qu'il possède dans sa table, il actualise sa table.
- s'il découvre une route vers un réseau connu mais plus longue, il ignore cette route.
- si le réseau n'évolue pas (panne ou ajout de nouveau matériel), les tables de routage convergent vers une valeur stable. Elles n'évoluent plus.
- si un routeur ne reçoit pas pendant 3 minutes d'information de la part d'un routeur qui lui avait auparavant communiqué sa table de routage, ce routeur est considéré comme en panne, et toutes les routes passant par lui sont affectées de la distance infinie. (16)

**Propriété 1: Optimisation en protocole RIP**

Le protocole RIP cherche à minimiser le nombre de sauts, donc le nombre de routeurs traversés.
En extrapolant un peu, on peut dire que RIP minimise la distance parcourue.

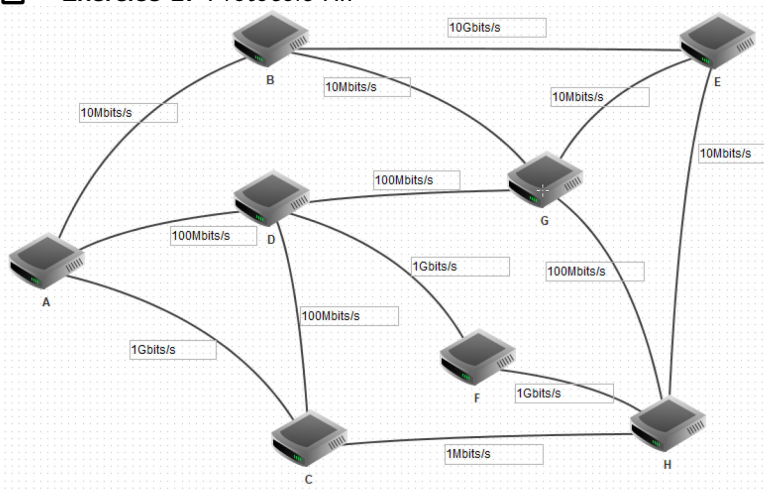
**Remarque 1: Notion de distance**

**Attention**: La *distance* (ou nombre de sauts) peut être considérée comme le nombre d'arêtes ou comme le nombre de routeurs traversés (dans ce cas, il faut retrancher 1 ). Comme souvent, vous pourrez rencontrer les 2 conventions. La plus commune est de compter le nombre de routeur traversés.

**Remarque 2: Remarques et inconvénients**

- Le protocole RIP n'admet qu'une distance maximale égale à 15 (ceci explique que 16 soit considéré comme la distance infinie), ce qui le limite aux réseaux de petite taille.
- Chaque routeur n'a jamais connaissance de la topologie du réseau tout entier: il ne le connait que par ce que les autres routeurs lui ont raconté. On dit que ce protocole de routage est du "Routing by rumor".
- La métrique utilisée (le nombre de sauts) ne tient pas compte de la qualité de la liaison, contrairement au protocole OSPF.

**Exercice 2:** Protocole RIP




- donner en protocole RIP la table de routage du routeur E

| Dest. | Routeur suivant | Métrique |
|---|---|---|
| A | B | 2 |
| B | | |
| C | | |
| D | | |
| F | | |
| G | | |
| H | | |

- déterminer les chemins possibles pour aller de A à E

# 4. Protocole OSPF

Un inconvénient majeur du protocole précédent est la non-prise en compte de la bande passante reliant les routeurs.

**Propriété 2: Principe fondamental du protocole OSPF**

Le chemin le plus rapide n'est pas forcément le plus court.

**Définition 5: Le protocole OSPF**

OSPF fait partie des protocoles à état de liens, c'est-à-dire qu'il prend en compte la qualité des liens pour déterminer la meilleure route.

Chaque routeur construit sa table de routage avec la métrique, c'est-à-dire les coûts des liaisons (temps de transmission, débit). Plus le coût est faible, plus la liaison sera favorisée.

On définit le coût par l'expression:

$$\text{cout} = \frac{10^8}{\text{débit en Mbit/s}}$$

**Propriété 3: Optimisation en protocole RIP**

Le protocole OSPF cherche à minimiser le temps de parcours, en choisissant les lignes les plus rapides.

**Exercice 3:** On utilisera dans cet exercice le même réseau que dans l'exercice 2

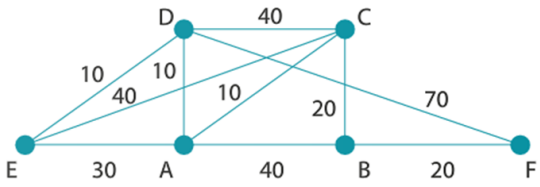
- déterminer les coûts des différentes liaisons
- compléter la table de routage du routeur F en OSPF
- indiquer le chemin emprunté par des informations entre le routeur E et le routeur D

| Dest. | Routeur suivant | Coût total |
|---|---|---|
| A | | |
| B | | |
| C | | |
| D | | |
| E | | |
| G | | |
| H | | |

## 5. Algorithme de Dijkstra

Dans le cas d'un graphe pondéré complexe, il faut trouver un algorithme de détermination du plus court chemin d'un point à un autre.
Edsger Dijkstra en 1959 a découvert cet algorithme qui porte son nombre
Vous pouvez regarder cette vidéo: `https://www.youtube.com/watch?v=rI-Rc7eF4iw`

**Exercice 4:** Algorithme de Dijkstra




| E | A | B | C | D | F | Choix |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

# ARSE04 – Sécurisation des communications

**Définition 1: Notion de chiffrement**

Afin de chiffrer des données, on est amené à utiliser une clé de chiffrement. En connaissant cette clé, il est évidemment possible de déchiffrer les données.

## 1. Histoire du chiffrement

**Propriété 1: Le code César**

Le principe est simple, il consiste à décaler les lettres dans l'alphabet. Ce chiffrement peut être cassé par:

- force brute: on teste toutes les clés possibles.
- analyse fréquentielle: on suppose qu'en français la lettre la plus fréquente est le `E`. Cela nous donne donc le décalage utilisé.

**Propriété 2: Chiffrement par substitution**

Cette technique consiste à substituer une lettre par une autre dans un alphabet réordonné qui sert de clé.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
WXCVBNMLKJUIOPHGFYTRDSQAEZ
```

Dans cet exemple, la lettre A est remplacée par la lettre W et ainsi de suite...

**Propriété 3: Chiffrement de Vigenere**

C'est une amélioration du code César. On fait varier le décalage pour chaque lettre de l'alphabet. Les décalages dont inscrits dans la clé qui peut être un mot, une phrase ou n'importe quel texte.

Exemple avec la clé `BAC`

```
Alphabet A B C D E F G H I J K L ...
Clé      B A C B A C B A C B A C ...
Décalage  1 0 2 1 0 2 1 0 2 1 0 2 ...
Codage   B B E E E H H H K K K N ...
```

**Propriété 4: Le chiffrement XOR**

L'opérateur `XOR` ou `OU Exclusif` est noté $\oplus$ est un opérateur dont la table de vérité est:

| X | Y | S = X ⊕ Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Table de vérité de XOR

L'opérateur `XOR` est commutatif: $X \oplus Y = Y \oplus X$ et possède la propriété suivant: $X \oplus Y \oplus Y = X$.
On peut donc revenir au message initial en appliquant deux fois la fonction `XOR`.

Étant donné un message et une clé de chiffrement, par exemple `NSI`:

```
UN MESSAGE TRÈS SECRET
NSINSINSINSINSINSINSIN
```

Chaque caractère du message est associé à une valeur numérique entière (par exemple son Unicode):

```
85 78 32 77 69 83 83 65 71 69 32 84 82 20 83 32 83 69 67 82 69 84
78 83 73 78 83 73 78 83 73 78 83 73 78 83 73 78 83 73 78 83 73 78
```

On effectue ensuite l'opération $\oplus$ entre chaque nombre du message et de la clé. Par exemple pour le premier caractère:

$$\begin{array}{rcccccccc} & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \quad 85 \\ \oplus & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \quad 78 \\ \hline & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \quad 27 \end{array}$$

Le code obtenu dans notre exemple est donc: `27 29 105 3 22 26 29 18 14 11 115 29 28 155 26 ...`

L'opérateur $\oplus$ étant réversible, la réitération de l'opération sur le message chiffré rendra le message original.

**Exercice 1:** Implémenter le chiffrement XOR en python

## 2. Chiffrement symétrique

**Définition 2: Principe du chiffrement symétrique**

Dans un chiffrement symétrique, c'est la même clé qui va servir au chiffrement et au déchiffrement.


**Définition 3: Notion de clé**

La clé est un renseignement permettant de chiffrer ou déchiffrer un message. Cela peut être:

- un nombre (dans un simple décalage des lettres de l'alphabet, comme le chiffre de César)
- une phrase (dans la méthode du masque jetable XOR)
- une image (imaginez un chiffrement où on effectue un XOR par les pixels d'une image)

Un chiffrement est dit symétrique lorsque la connaissance de la clé ayant servi au chiffrement permet de déchiffrer le message.

**Propriété 5: Avantages et inconvénients d'un chiffrement symétrique**

| `Avantages` | `Inconvénients` |
|---|---|
| Les chiffrements symétriques sont souvent rapides, consommant peu de ressources et donc adaptés au chiffrement de flux important d'informations.<br>La sécurisation des données transitant par le protocole `https` est basée sur un chiffrement symétrique. | La clé! Si Alice et Bob ont besoin d'utiliser un chiffrement pour se parler, cela impose qu'Alice et Bob aient pu établir une connexion sécurisée pour s'échanger cette clé. |

**Remarque 3: Un chiffrement symétrique est-il un chiffrement de mauvaise qualité?**

Pas du tout! S'il est associé naturellement à des chiffrements simples et faibles (comme le décalage de César), un chiffrement symétrique peut être très robuste... voire inviolable.

C'est le cas du masque jetable. Si le masque avec lequel on effectue le XOR sur le message est aussi long que le message, alors il est impossible de retrouver le message initial. Pourquoi?

Imaginons qu'Alice veuille transmettre le message clair "LUNDI". Elle le chiffre avec un masque jetable (que connait aussi Bob), et Bob reçoit donc "KHZOK". Si Marc a intercepté le message "KHZOK", même s'il sait que la méthode de chiffrement utilisée est celle du masque jetable (principe de Kerckhoffs), il n'a pas d'autre choix que de tester tous les masques de 5 lettres possibles.
Ce qui lui donne $26^5$ possibilités (plus de 11 millions) pour le masque, et par conséquent $26^5$ possibilités pour le message déchiffré...
Cela signifie que Marc verra apparaitre, dans sa tentative de déchiffrage, les mots "MARDI", "JEUDI", "JOUDI", "STYLO", "FSDJK", "LUNDI", "LUNDA"... Il n'a aucune possibilité de savoir où est le bon message original parmi toutes les propositions (on parle de sécurité sémantique).

**Propriété 6: Principe de Kerckhoffs**

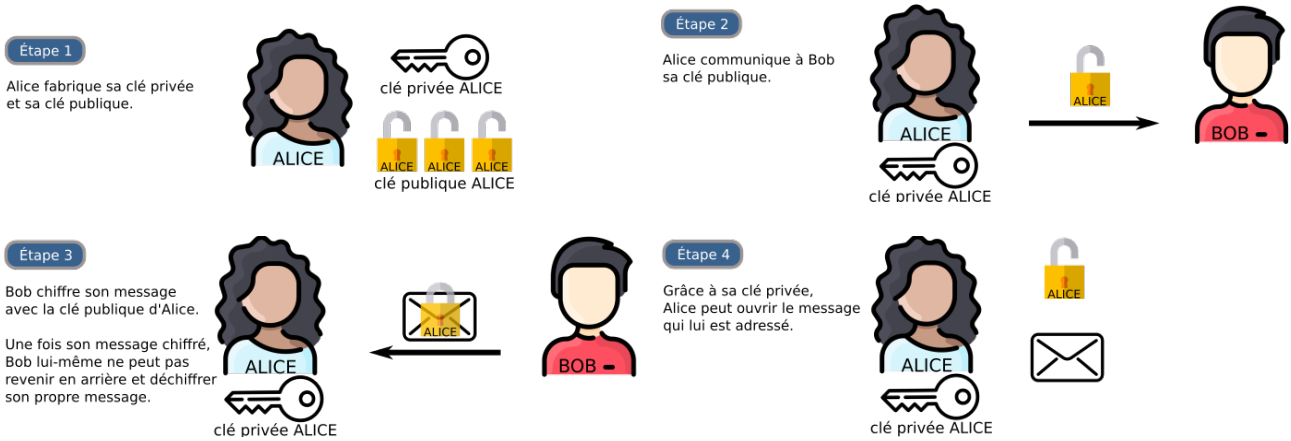
La sécurité d'un système de chiffrement ne doit reposer que sur le secret de la clé, et non pas sur la connaissance de l'algorithme de chiffrement. Cet algorithme peut même être public (ce qui est pratiquement toujours le cas).

# 3. Chiffrement Asymétrique

**Définition 4: Principe du chiffrement asymétrique**

Dans un **chiffrement asymétrique**, deux clés différentes sont utilisées:

- Une **clé publique**, qui est connue de tous et sert à chiffrer les messages.
- Une **clé privée**, qui est gardée secrète et sert à déchiffrer les messages.




**Propriété 7: Avantages et inconvénients d'un chiffrement asymétrique**

`Avantages`

- Pas besoin d'échanger une clé secrète au préalable.
- Permet de garantir la confidentialité et l'authenticité des messages.

`Inconvénients`

- Plus lent que le chiffrement symétrique.
- Nécessite des calculs complexes, donc plus gourmand en ressources.

**Propriété 8: Le rôle interchangeable des clés publiques et privées**

L'illustration précédente associe:

- une image de cadenas à la clé publique (car on s'en sert pour chiffrer les messages)
- une image de clé à la clé privée (car on s'en sert pour déchiffrer les messages)

Concrètement, la clé privée et la clé publique sont deux nombres aux rôles identiques. Appelons-les A et B:

- il est impossible de trouver A en fonction de B. Réciproquement, si on connait A, il est impossible d'en déduire B.
- si on chiffre un message avec A, on peut le déchiffrer avec B. Réciproquement, si on chiffre avec B, on peut déchiffrer le message grâce à A.
- on peut donc chiffrer avec une clé publique et déchiffrer avec la clé privée associée (ce qui est fait dans l'exemple précédent). Mais on peut aussi chiffrer avec la clé privée, et déchiffrer avec la clé publique associée.

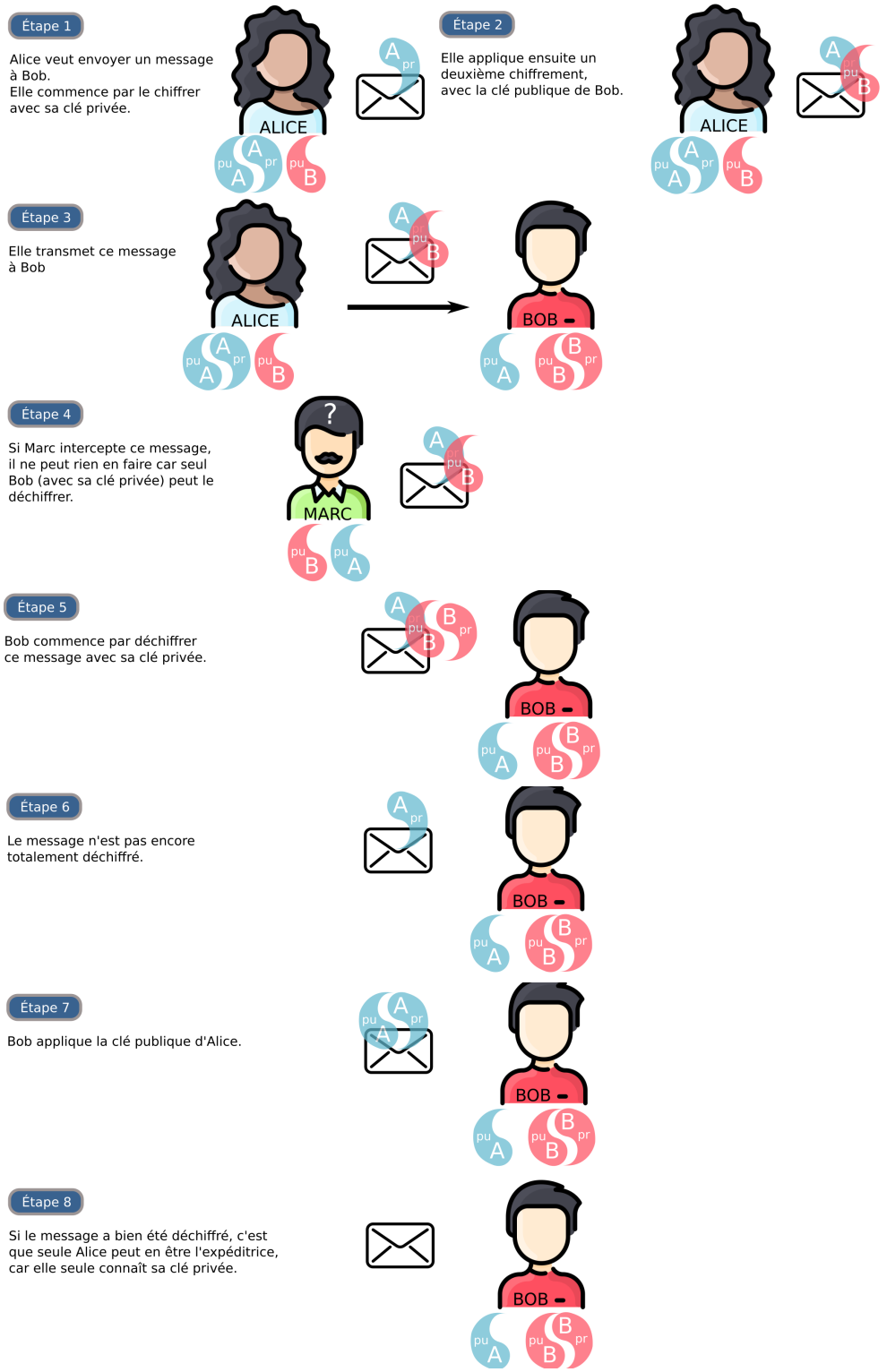
A et B ont donc des rôles interchangeables (chacun peut être un cadenas, chacun peut être une clé), et ce n'est qu'en connaissant A et B qu'on peut déchiffrer le message.

**Exemple 1: Conventions**

## 4. Communications authentifiées

Dans cette version du chiffrement asymétrique, Alice (qui a distribué largement sa clé publique) ne peut pas s'assurer que le message vient bien de Bob. Il peut avoir été créé par Marc, qui signe "Bob" et usurpe ainsi son identité.
Le protocole que nous allons décrire ci-dessous permet:

- d'empêcher qu'un message intercepté soit déchiffré (ce qui était déjà le cas dans le chiffrement asymétrique)
- mais aussi de s'assurer que chaque personne est bien celle qu'elle prétend être: on résout le "problème d'authentification".

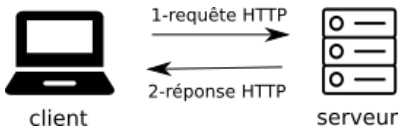



- Alice est sûre que seul Bob pourra déchiffrer le message qu'elle envoie.
- Bob est sûr que le message qu'il reçoit vient bien d'Alice.

# 5. HTTPS: exemple d'utilisation conjointe d'un chiffrement asymétrique et d'un chiffrement symétrique.

**Définition 5: Le protocole HTTP**

Avant de parler du protocole HTTPS, petit retour sur le protocole HTTP: un client effectue une requête HTTP vers un serveur, le serveur va alors répondre à cette requête (par exemple en envoyant une page HTML au client).




**Propriété 9: Incovénients du protocole HTTPS**

- Un individu qui intercepterait les données transitant entre le client et le serveur pourrait les lire sans aucun problème.
- Grâce à une technique de `DNS Spoofing`, un serveur "pirate" peut se faire passer pour un site sur lequel vous faites par exemple des achats.

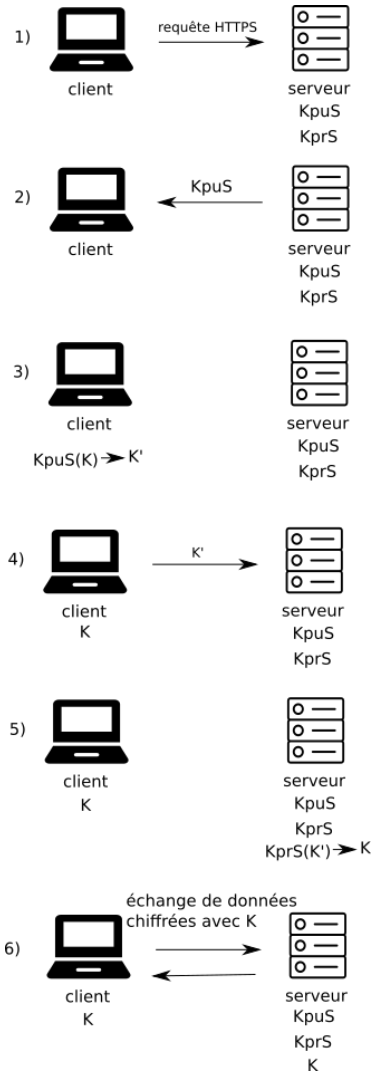
**Définition 6: Le protocole HTTPS**

HTTPS est donc la version sécurisée de HTTP, le but de HTTPS est d'éviter les 2 problèmes évoqués ci-dessus. HTTPS s'appuie sur le protocole TSL (Transport Layer Security) anciennement connu sous le nom de SSL (Secure Sockets Layer).

**Propriété 10: Fonctionnement du protocole HTTPS**

- le client effectue une requête HTTPS vers le serveur, en retour le serveur envoie sa clé publique (KpuS) au client
- le client "fabrique" une clé K (qui sera utilisé pour chiffrer les futurs échanges), chiffre cette clé K avec KpuS et envoie la version chiffrée de la clé K au serveur
- le serveur reçoit la version chiffrée de la clé K et la déchiffre en utilisant sa clé privée (KprS). À partir de ce moment-là, le client et le serveur sont en possession de la clé K
- le client et le serveur commencent à échanger des données en les chiffrant et en les déchiffrant à l'aide de la clé K (chiffrement symétrique).

Ce processus se répète à chaque fois qu'un nouveau client effectue une requête HTTPS vers le serveur.




Pour éviter tout problème, il faut que le serveur puisse justifier de son "identité". Pour ce faire, chaque site désirant proposer des transactions HTTPS doit, périodiquement, demander (acheter dans la plupart des cas) un certificat d'authentification (sorte de carte d'identité pour un site internet) auprès d'une autorité habilitée à fournir ce genre de certificats (chaque navigateur web possède une liste des autorités dont il accepte les certificats).

En cas d'absence de certificat (ou d'envoi de certificat non conforme), le client stoppe immédiatement les échanges avec le serveur.

Dans ce cas, le navigateur web coté client affichera une page de mise en garde avec un message du style "ATTENTION le certificat d'authentification du site `XXXX` a expiré, il serait prudent de ne pas poursuivre vos échanges avec le site `XXXX`".