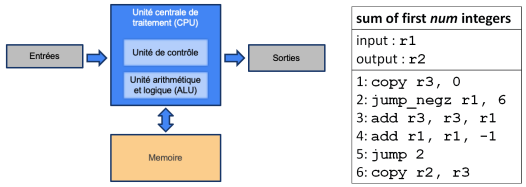
Information, Calcul et Communication

CS-119(g) ICC – Théorie
Semaine 03

Rafael Pires
rafael.pires@epfl.ch

Lausanne, 26.09.2025

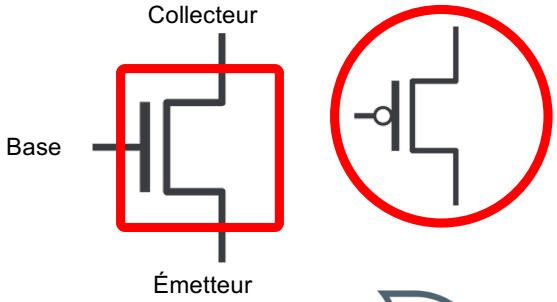
Précédemment, dans ICC-T 02

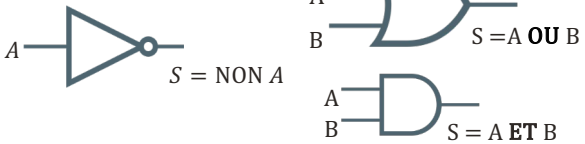


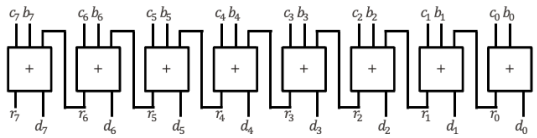
sum of first *num* integers

input : r1
output : r2

```
1: copy r3, 0
2: jump_negz r1, 6
3: add r3, r3, r1
4: add r1, r1, -1
5: jump 2
6: copy r2, r3
```







- **L'architecture de von Neumann** : mémoire, processeur, instructions et flux de données.
- **Le transistor** : base de toute l'électronique numérique, agit comme un interrupteur.
- **Portes logiques** : associations de transistors pour effectuer des opérations binaires (ET, OU, NON…)
- **Circuits logiques** : combinaisons de portes logiques qui réalisent des fonctions plus complexes, comme un **additionneur** 8-bits.
- **Assembleur** : instructions élémentaires (add, subtract, multiply) opérant sur des **registres**.

Programme du cours

Calcul

- Algorithmes
- Complexité
- Conception d'algorithmes
- Calculabilité
- **Circuits, architecture**

Information

- **Représentation de nombres**
- Echantillonnage et reconstruction de signaux
- Entropie
- Compression

Communication

- **Réseau**
- **Cryptographie**

Aujourd'hui

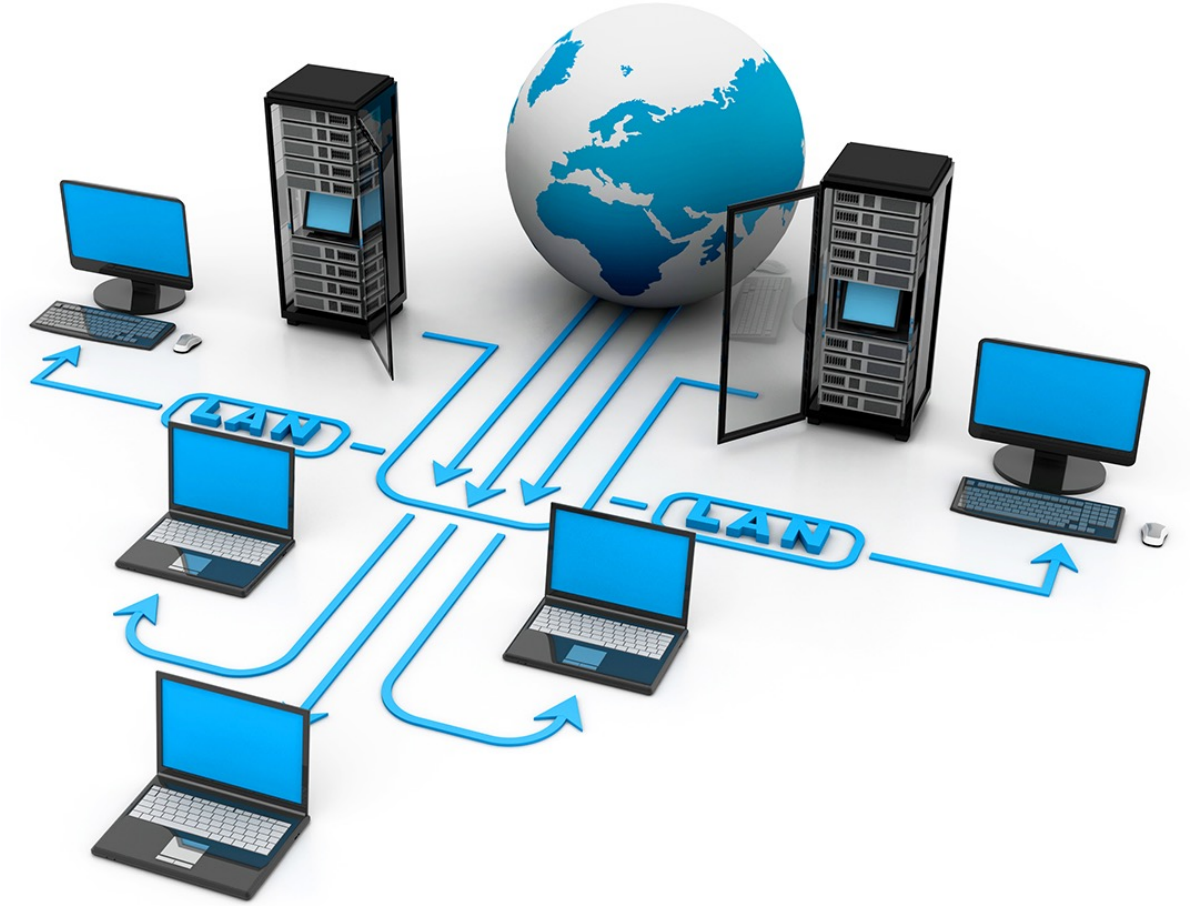
- **Introduction aux réseaux informatiques : paquets et couches de protocoles**
- Routage (Protocole IP)
  - Comment les données trouvent leur chemin (routage/IP)

- Introduction à la Cryptographie
  - Chiffrement symétrique
  - Protocole de Diffie-Hellman
  - Chiffrement asymétrique

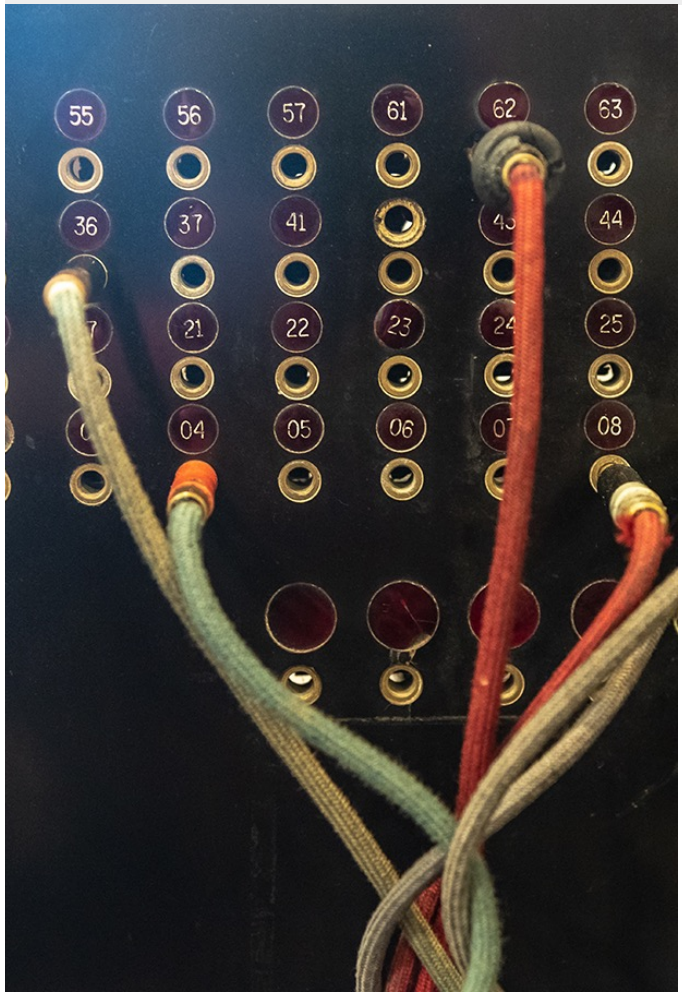
Introduction aux réseaux informatiques : paquets et couches de protocoles

- Explosion des réseaux : PC, smartphones, objets connectés
- Internet ≈ interconnexion de réseaux
- Internet ≠ Web !
- Objectifs du cours : paquets, protocoles, couches, routage

Réseaux : Internet

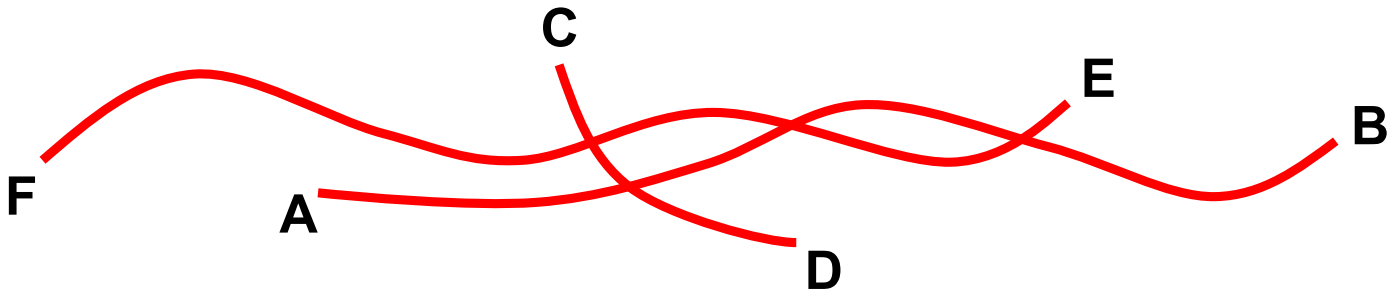


Réseau : Le téléphone



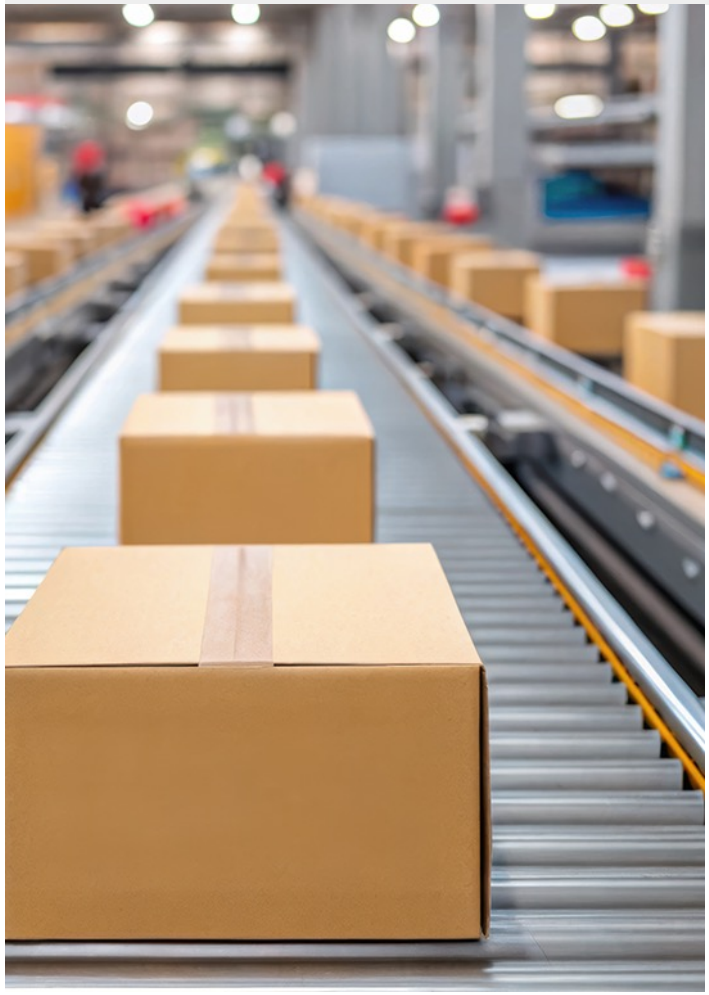
Principe :

- Établir une ligne de communication pour chaque paire d'utilisateurs désirant se parler
- Fermer celle-ci lorsque la communication est finie
  (et en ouvrir d'autres au fil des demandes)



Avec l'avènement des communications entre ordinateurs (**beaucoup plus de données** à échanger), ce système ne pouvait plus tenir...

Communication par paquets

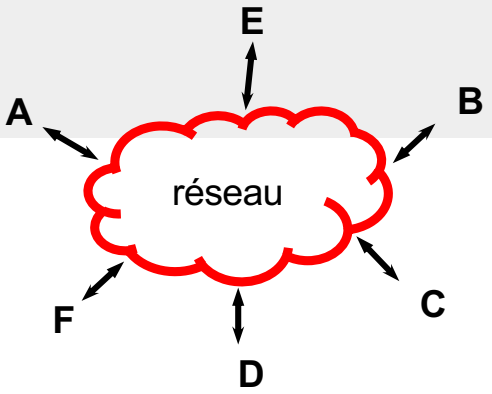


- Messages découpés en paquets
- Chaque paquet contient :
  - Données (payload)
  - En-tête (destination, numéro, etc.)
- Acheminement indépendant
- Besoin de fiabilité : erreurs, pertes, doublons

Structure d'un paquet

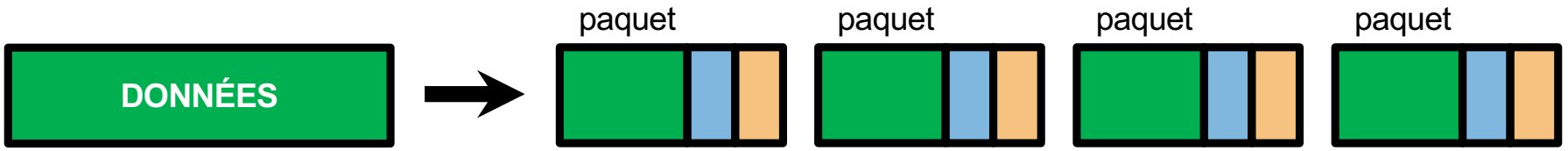
- En-tête + données (payload)
- Champs classiques :
  - Source, destination
  - Type, numéro de séquence
  - Contrôle d'erreur (checksum)

Réseau : Internet



Principe (schématique) de la communication par paquets :

Les données transmises par un utilisateur sont découpées
en **paquets**, qui sont ensuite envoyés dans le réseau :



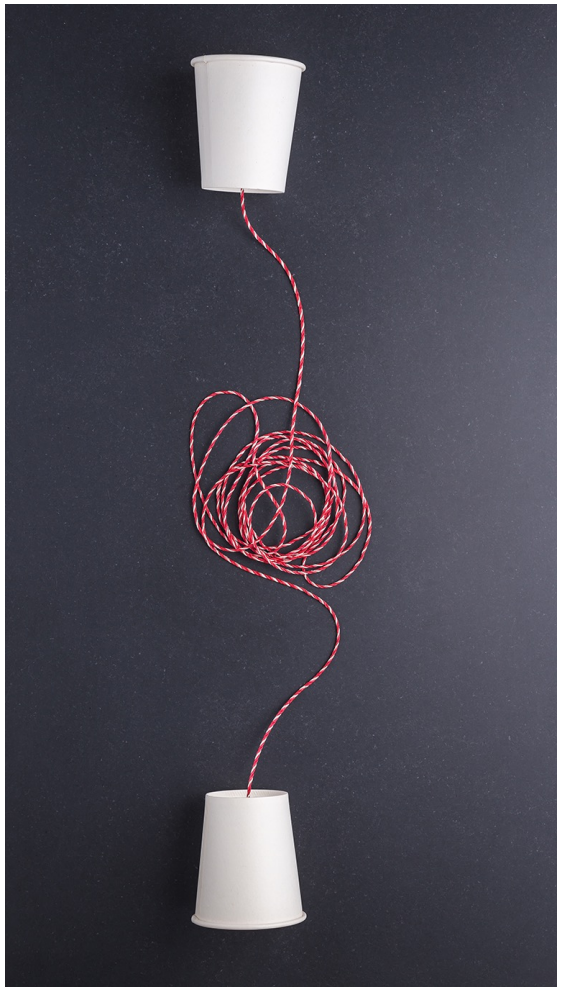
À chaque paquet on ajoute des *informations*, par exemple :

- Sa **destination**
- Son **identifiant**

(pour que le destinataire puisse remettre les paquets dans l'ordre à l'arrivée)

EPFL

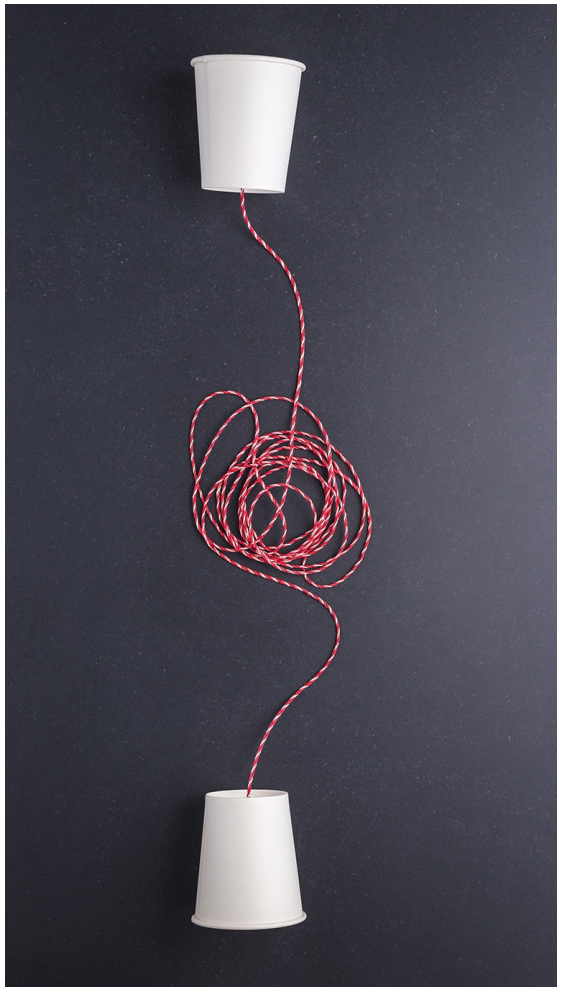
Notion de protocole



- Ensemble de règles pour communiquer

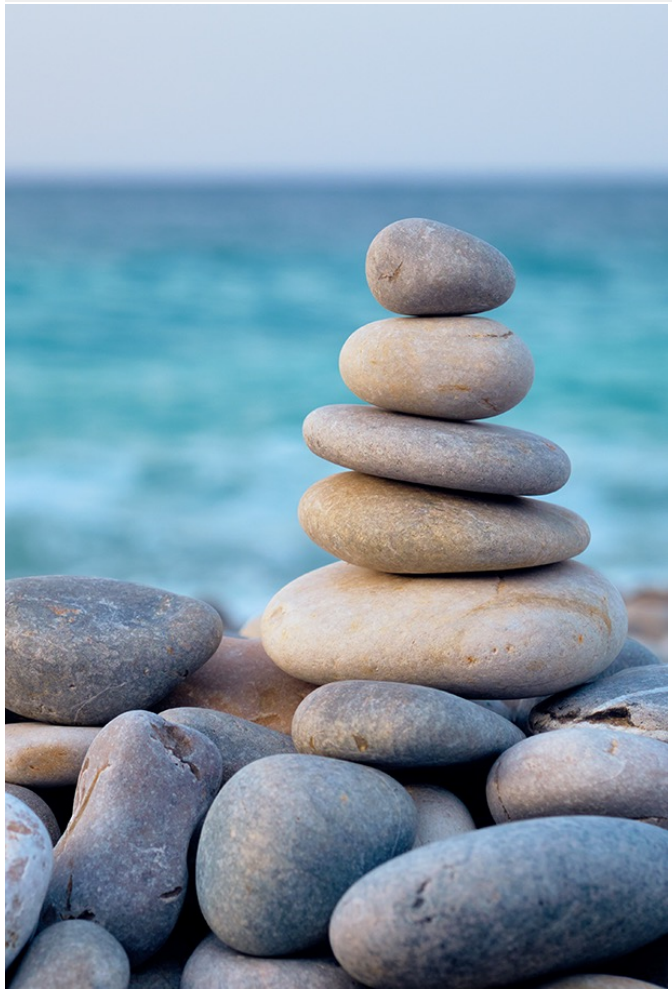
- Exemple humain : lever la main en classe
- Exemple informatique : ACK, retransmission, checksum

Téléphone à ficelle : un réseau… avec protocole !



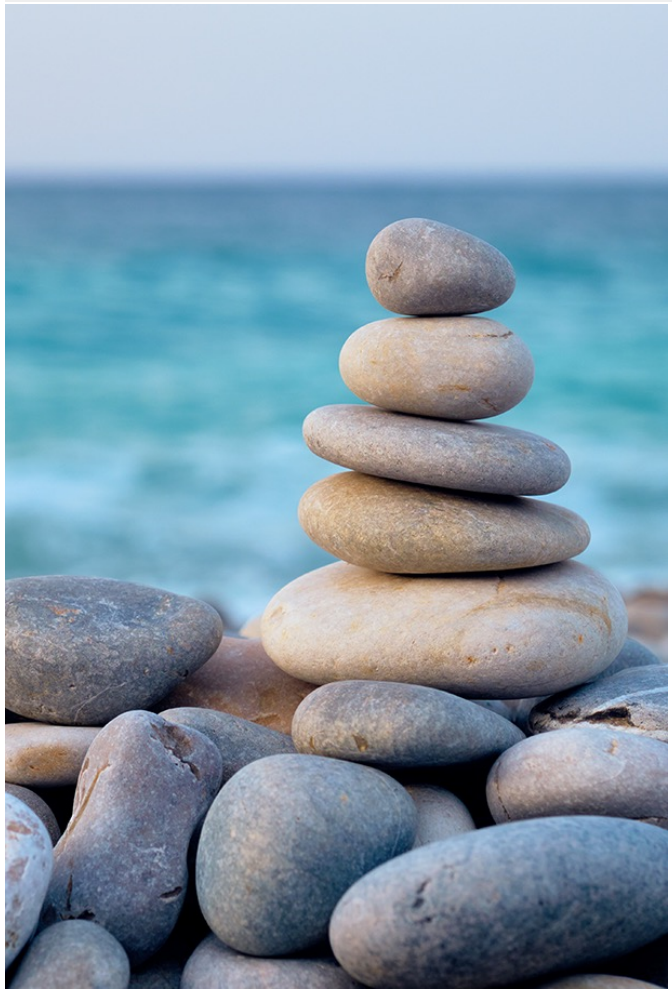
- Même avec deux gobelets reliés par une ficelle, on a besoin de règles pour que la communication fonctionne :
  - Accès au canal (couche liaison) : On ne peut pas parler tous les deux en même temps
    « *Tu parles,* je *t'écoute. Puis je réponds.* »
  - Détection d'erreur / demande de répétition si le son est étouffé
    « *Hein ? Tu peux répéter ?* »
  - Démarrage/fin de communication
    « *Allô ? T'es là ?* », « *J'ai fini !* »

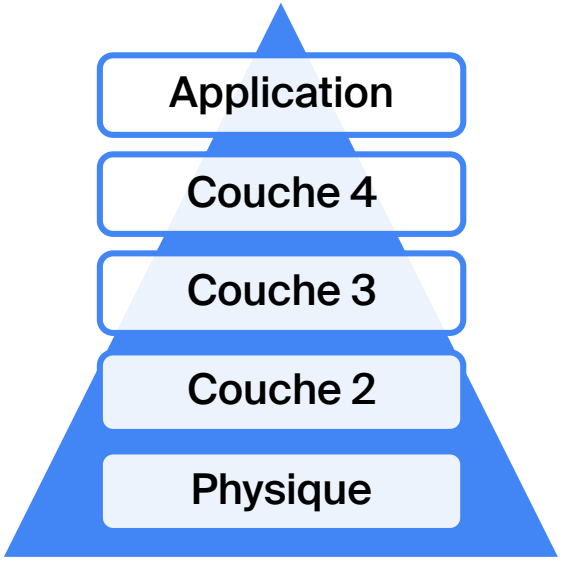
Pile de protocoles

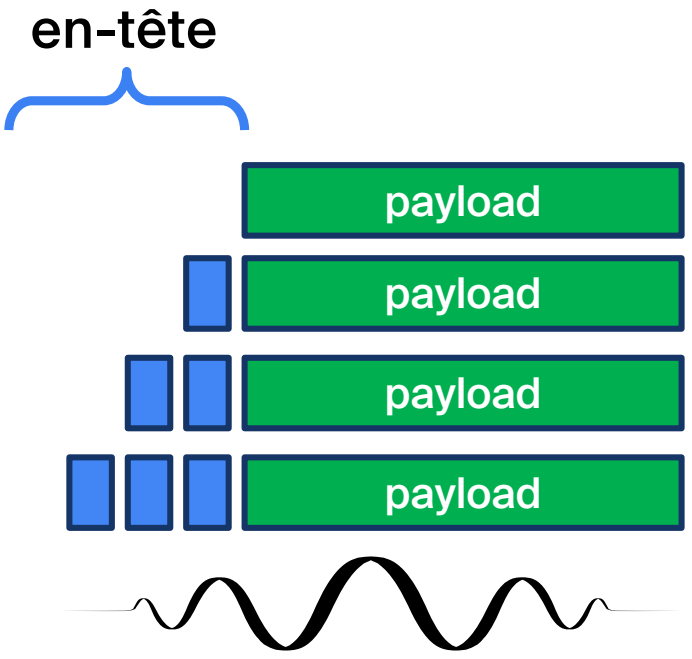


- Complexité → décomposition en couches
- Modèle TCP/IP (5 couches)
- Chaque couche ajoute un en-tête (encapsulation)
- Décapsulation à la réception

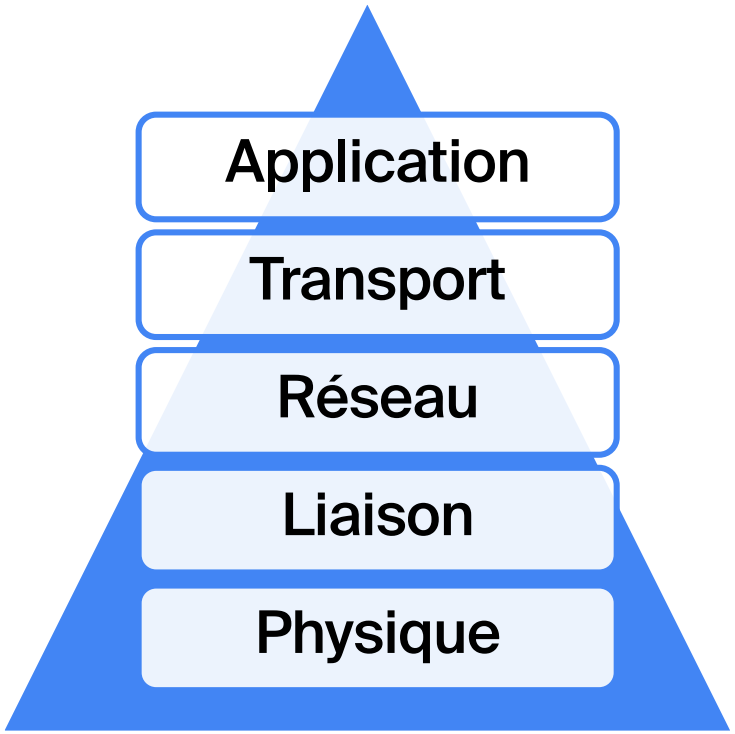
Pile de protocoles





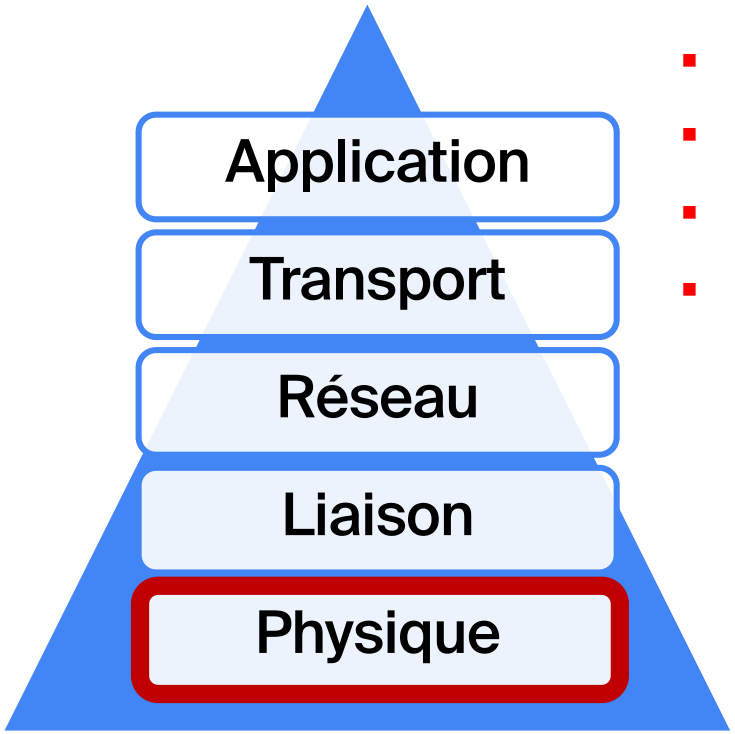


Les 5 couches TCP/IP

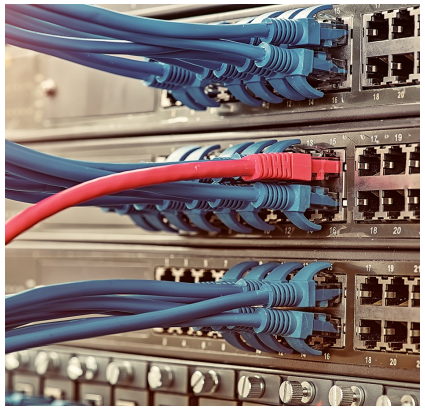


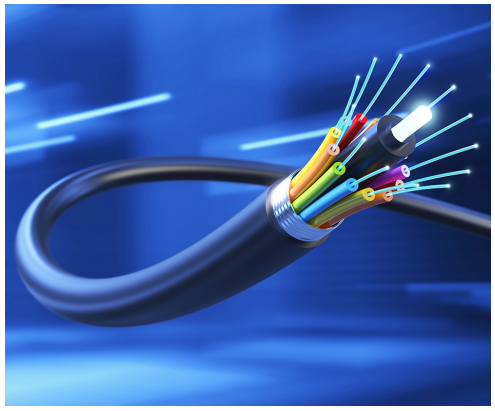
- Application (HTTP, mail, etc.)
- Transport (TCP, UDP)
- Réseau (IP)
- Liaison de données (Ethernet, Wi-Fi)
- Physique (câble, ondes)

Les 5 couches TCP/IP : Physique (couche 1)

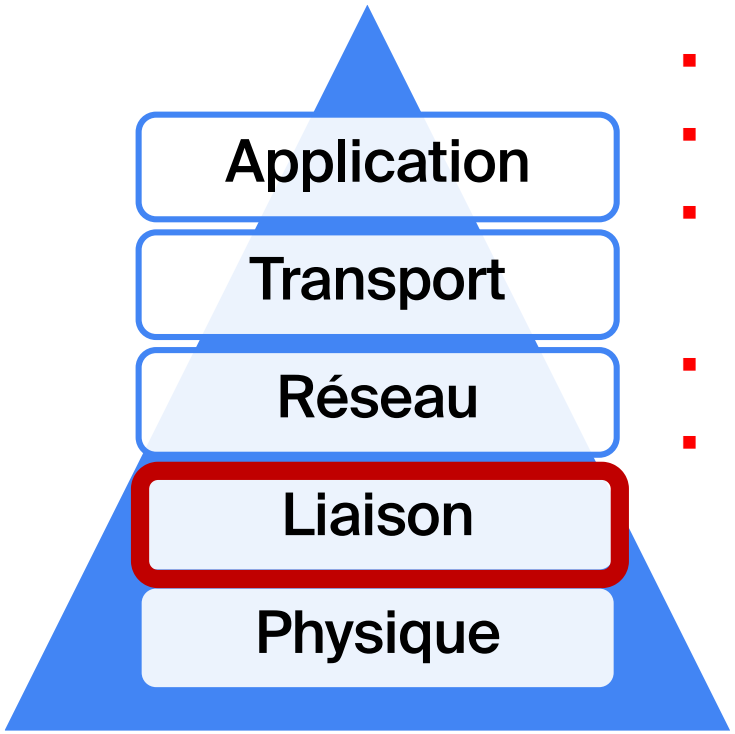


- Transmission brute des bits
- Support : ondes, fils, fibres optiques, radio...
- Pas d'en-tête : juste des signaux physiques (tension, lumière, ondes)
- Exemples : Ethernet à 1 Gbps, Wi-Fi, LTE, 5G, satellite, fibre optique

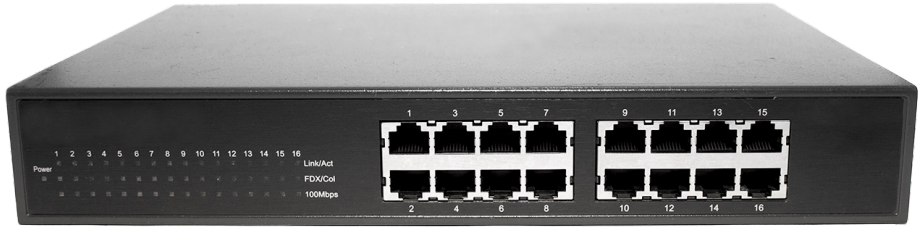




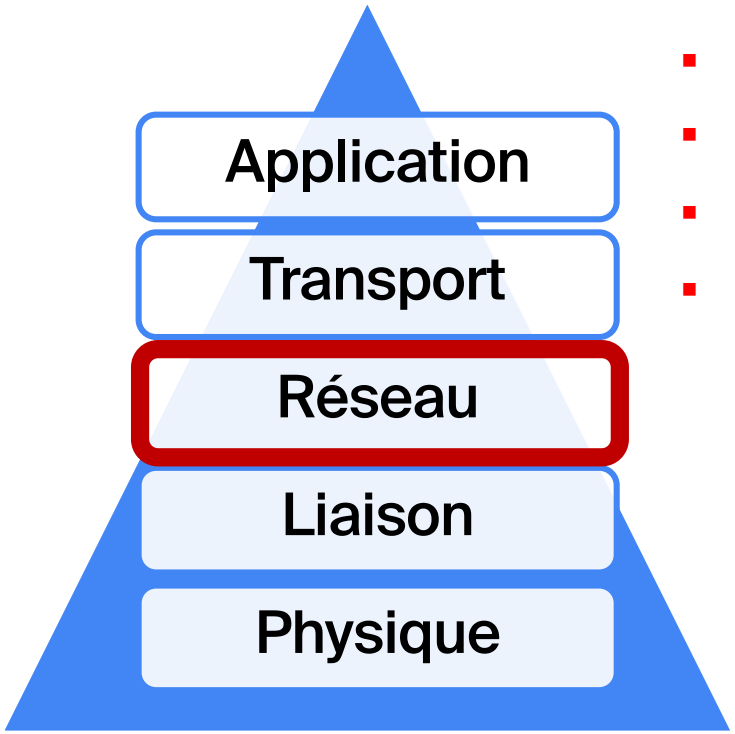
Les 5 couches TCP/IP : Liaison (couche 2)



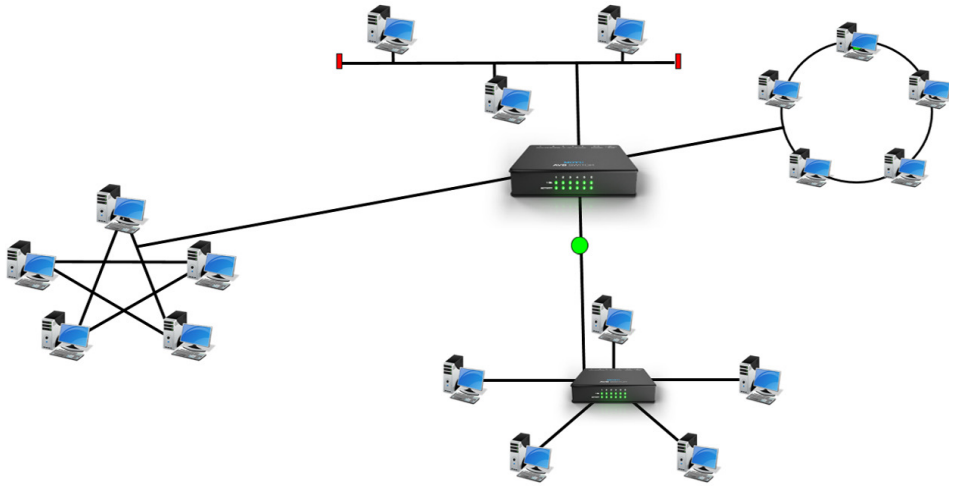
- Gère l'accès au canal (MAC), collisions, retransmissions locales
- Adresse locale : adresse MAC (ex.: `f8:f2:1e:aa:cd:c0`)
- Portée : concernée par la communication
  au sein d'un réseau local (LAN)
- Encapsulation : ajout d'un en-tête
- Exemples de protocoles : Ethernet, Wi-Fi (IEEE 802.11), TDMA, FDMA, CSMA/CD, PPP



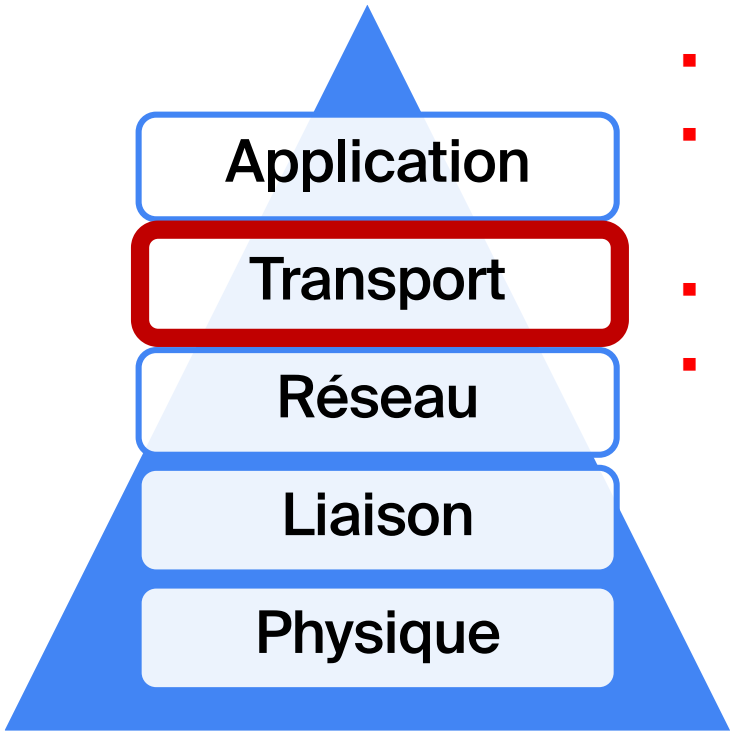
Les 5 couches TCP/IP : Réseau (couche 3)



- Routage des paquets de bout en bout (protocole IP)
- Adresse logique : adresse IP (ex. : `104.20.229.42`)
- En-tête IP avec adresses, TTL, etc.
- Rôle du routeur = lire l'en-tête et relayer

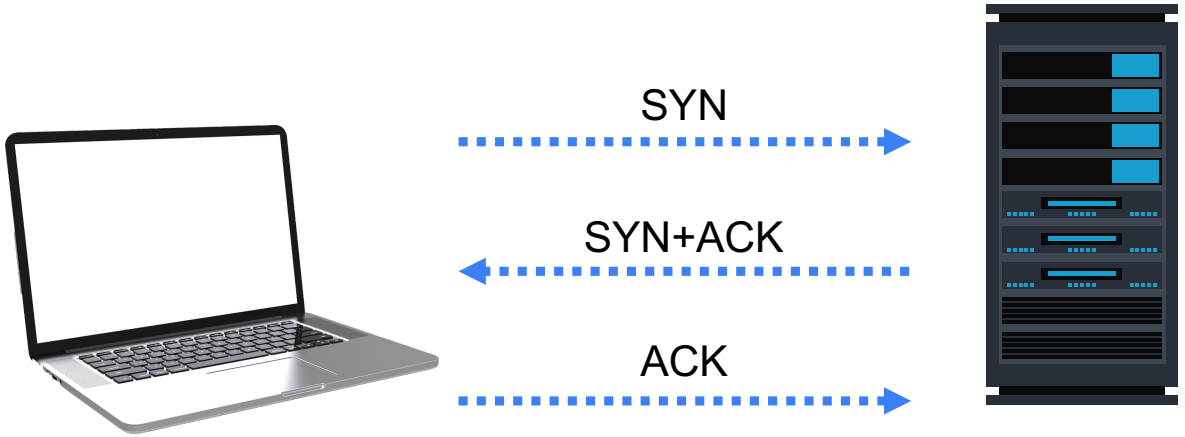


Les 5 couches TCP/IP : Transport (couche 4)

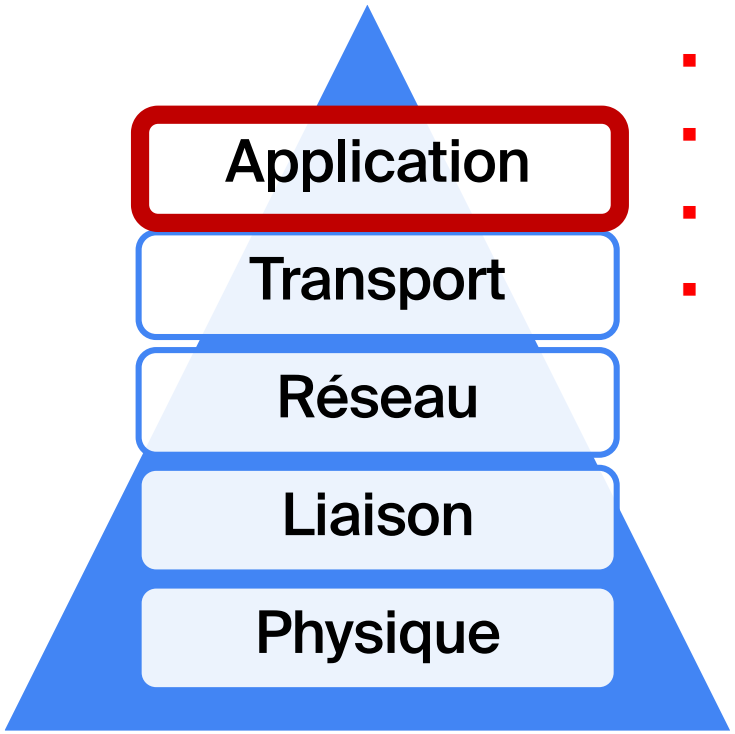


- Communication de processus à processus
- Adresse utilisée : numéro de port

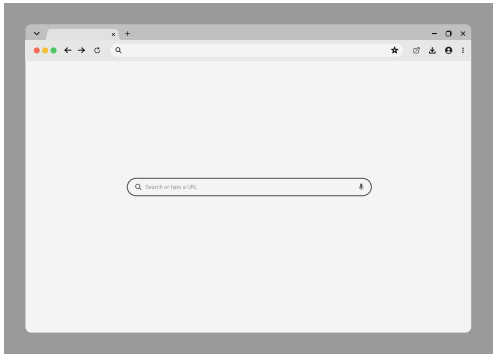
  identifie l'application cible (HTTP = 80, etc.)
- TCP : fiable, avec ACK, numérotation
- UDP : plus simple, sans vérification

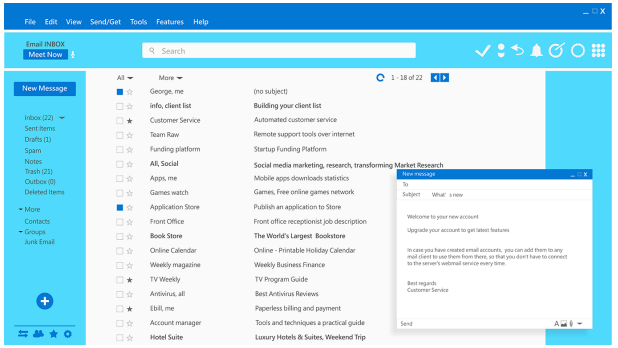


Les 5 couches TCP/IP : Application (couche 5)

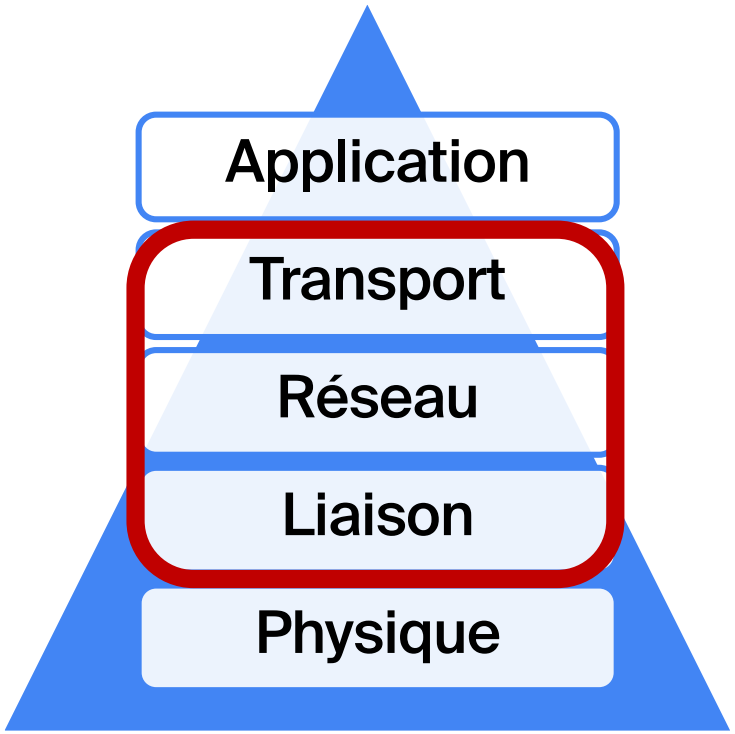


- Interface avec l'utilisateur
- Protocoles : HTTP, SMTP, FTP, DNS...
- Requêtes GET/POST, messages email, etc.
- Vue logique des échanges



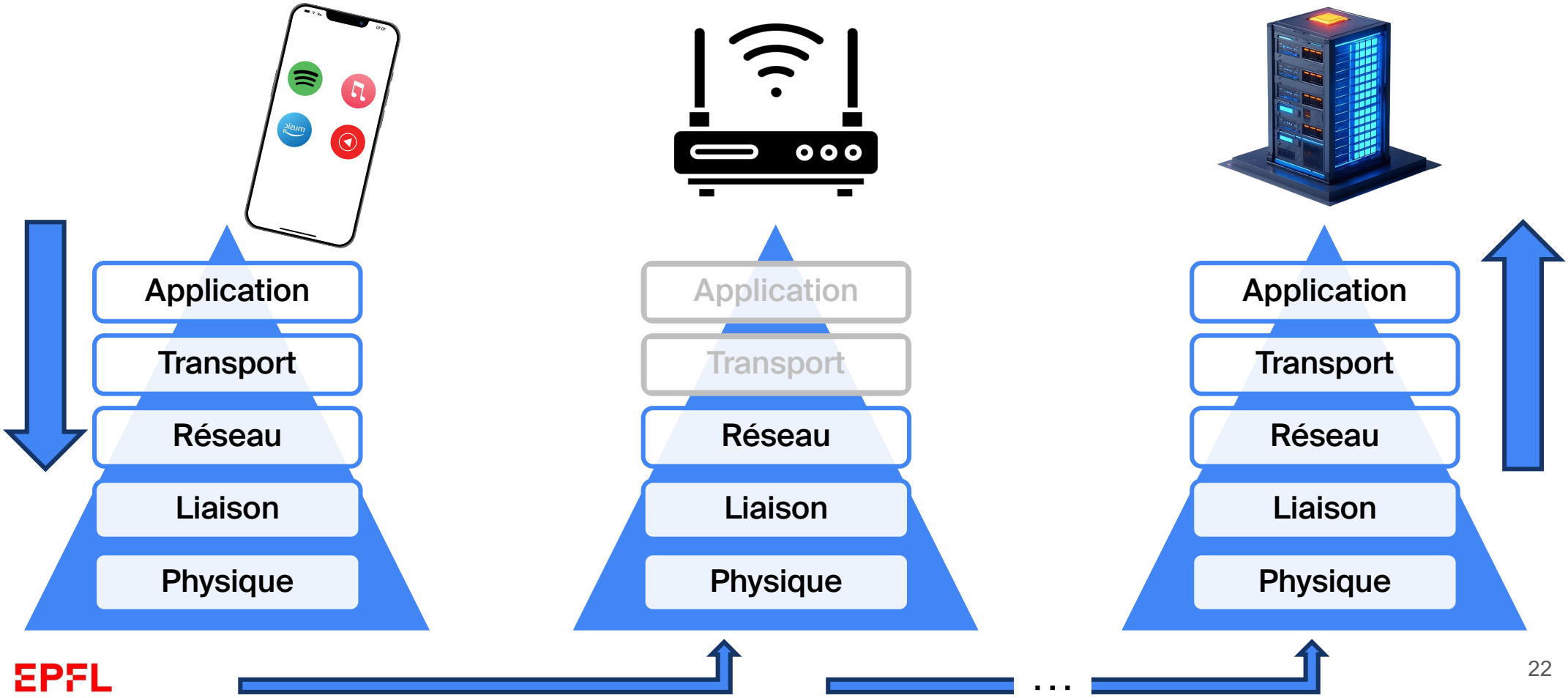


Les couches TCP/IP



Couche	Nom	Adresse	Portée
4	Transport	Port	Machine (processus)
3	Réseau	IP	Global (bout en bout)
2	Liaison	MAC	Local (réseau local)

Pile de protocoles



Aujourd'hui

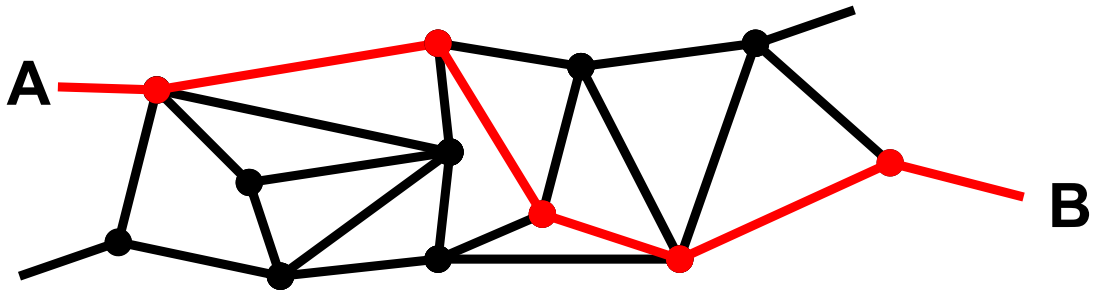
- Introduction aux réseaux informatiques :
  paquets et couches de protocoles
- **Routage (Protocole IP)**
  - **Comment les données trouvent leur chemin (routage/IP)**

- Introduction à la Cryptographie
  - Chiffrement symétrique
  - Protocole de Diffie-Hellman
  - Chiffrement asymétrique

Le protocole IP

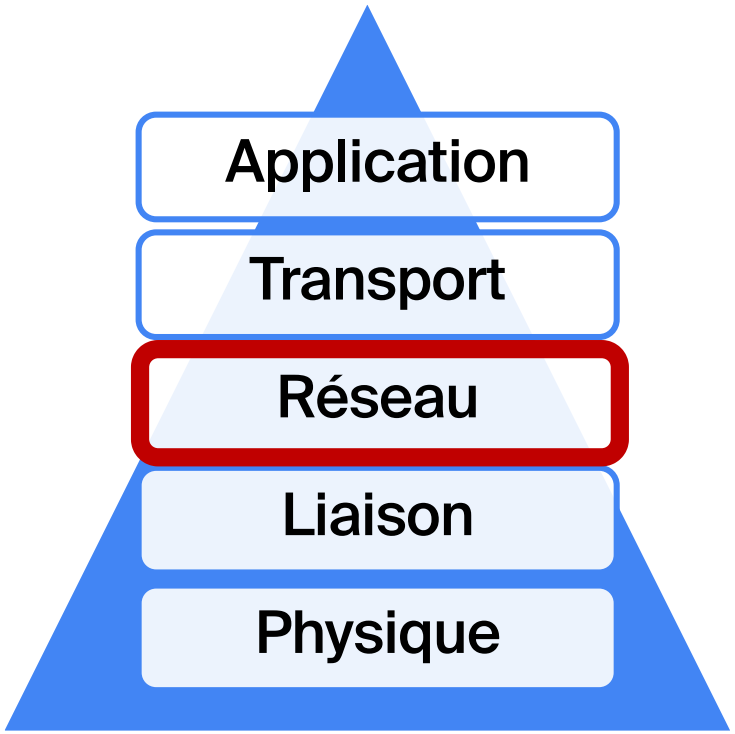
Nous allons maintenant étudier plus attentivement :

1. La question de **l'acheminement** du paquet à travers le réseau par un algorithme de **routage** :



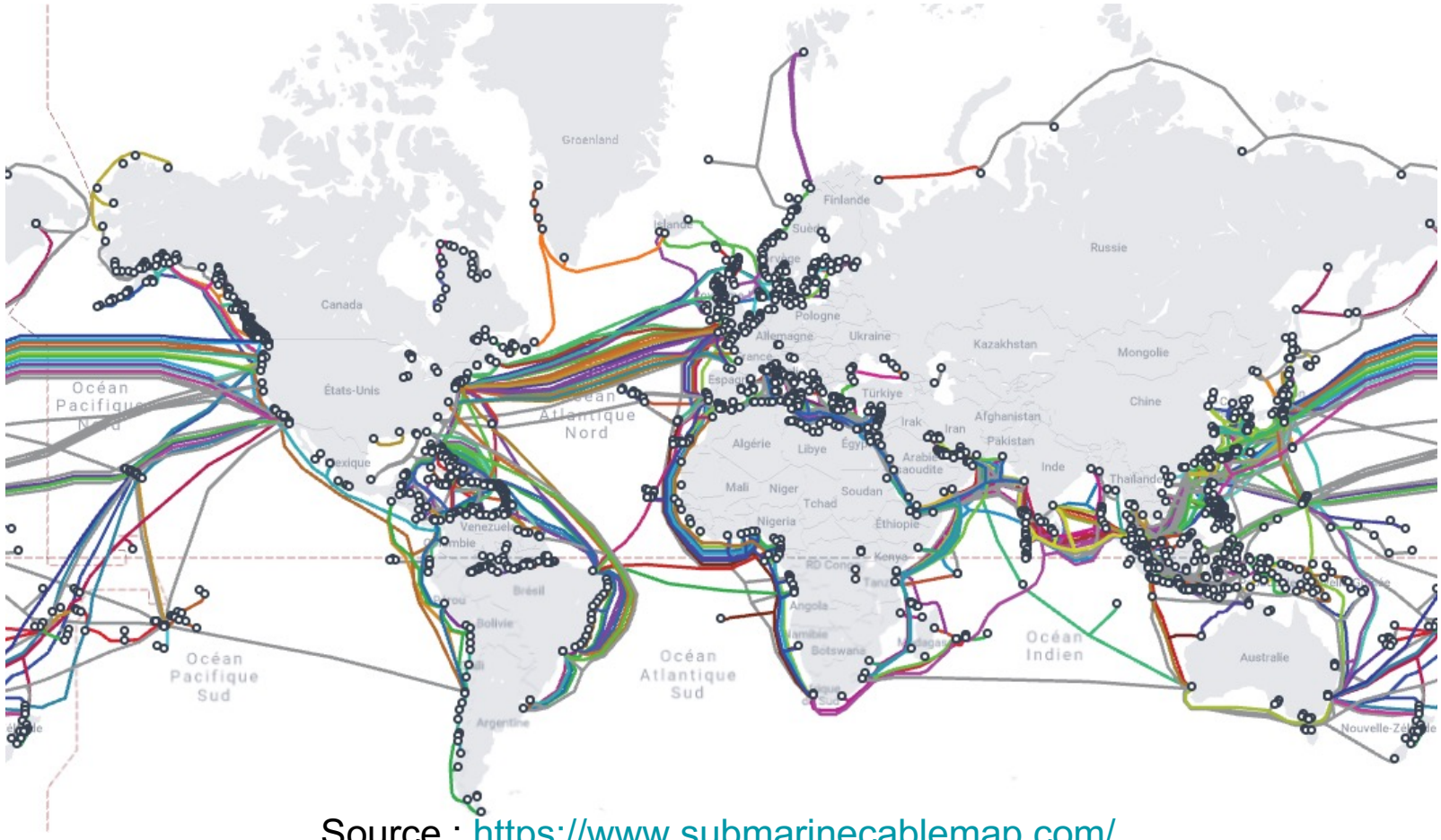
Protocole **IP**

Les 5 couches TCP/IP



- Application (HTTP, mail, etc.)
- Transport (TCP, UDP)
- Réseau (IP)
- Liaison de données (Ethernet, Wi-Fi)
- Physique (câble, ondes)

Routage - intuition



Source : https://www.submarinecablemap.com/

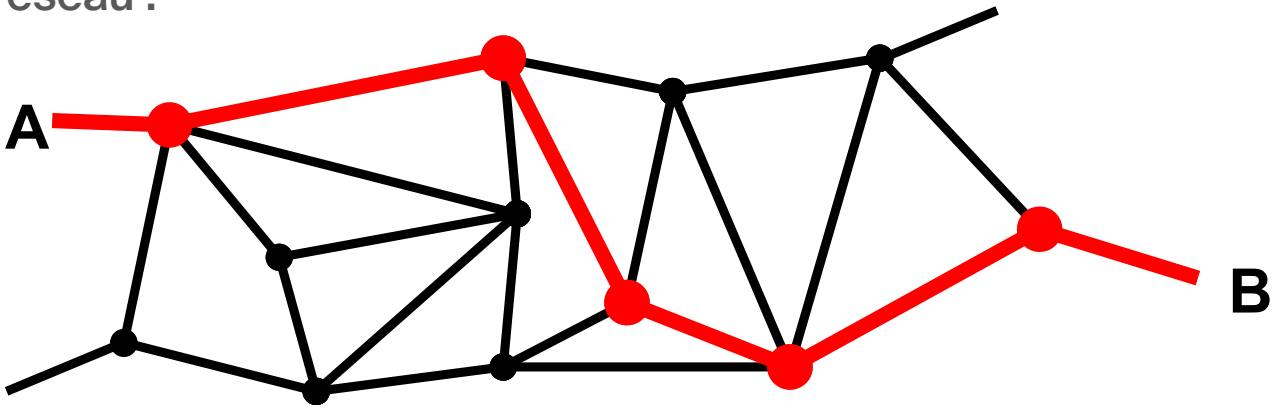
Routage - intuition

- Objectif : trouver le chemin de A à B
- Utilisation de graphes

  (nœuds = machines)
- Mesure : nombre de sauts (hops)

Protocole IP *(internet protocol)*

- Ce protocole est celui utilisé pour acheminer un paquet d'un nœud A vers un nœud B à travers le réseau :

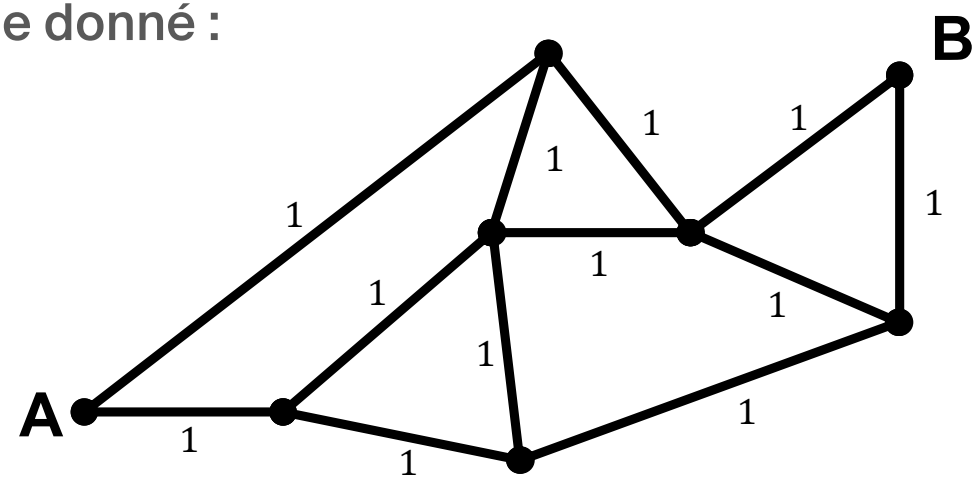


- Pour introduire le sujet, parlons d'abord du problème de **la recherche du plus court chemin dans un graphe**, ainsi que d'un algorithme possible pour la résolution de ce problème : l'algorithme BFS (Breadth First Search)

(« parcours en largeur »)

Algorithme BFS

Considérons un graphe donné :



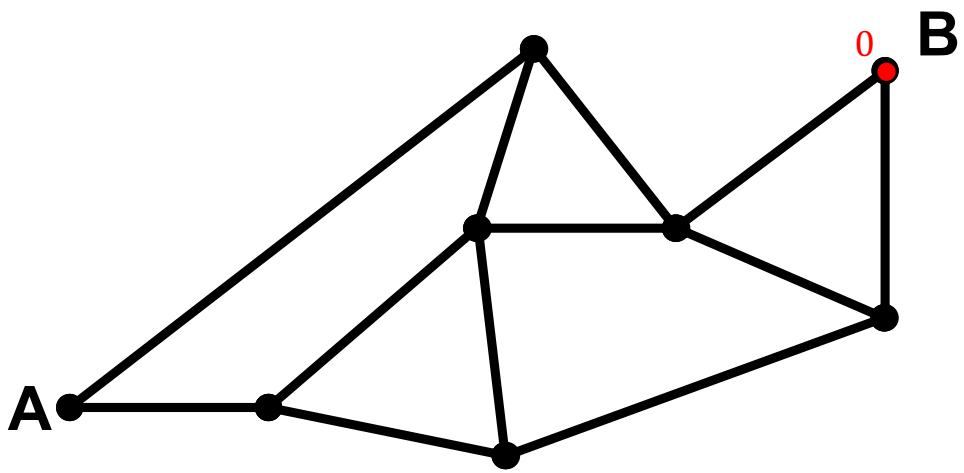
- Remarque :

Nous supposons ici que la distance entre deux nœuds directement reliés entre eux est toujours la même (disons 1).

Algorithme BFS

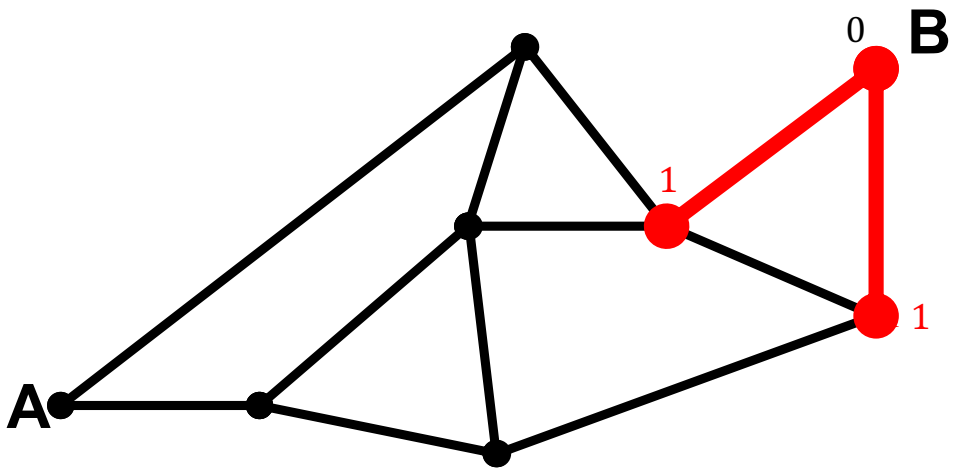
Pour trouver le chemin le plus court de A à B, l'algorithme BFS propose de chercher **tous les chemins les plus courts** de n'importe quel nœud du graphe au nœud B.

0. Initialisation : le nœud B est à distance 0 de lui-même.
   On marque celui-ci comme vu (pas besoin d'y revenir).



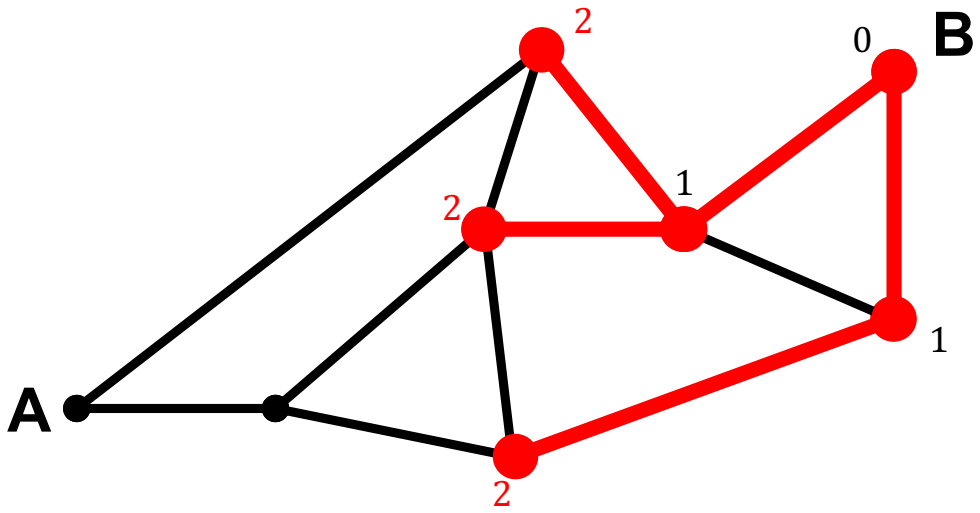
Algorithme BFS

1. Les **voisins directs** de B sont à distance 1 de celui-ci. On les marque également.



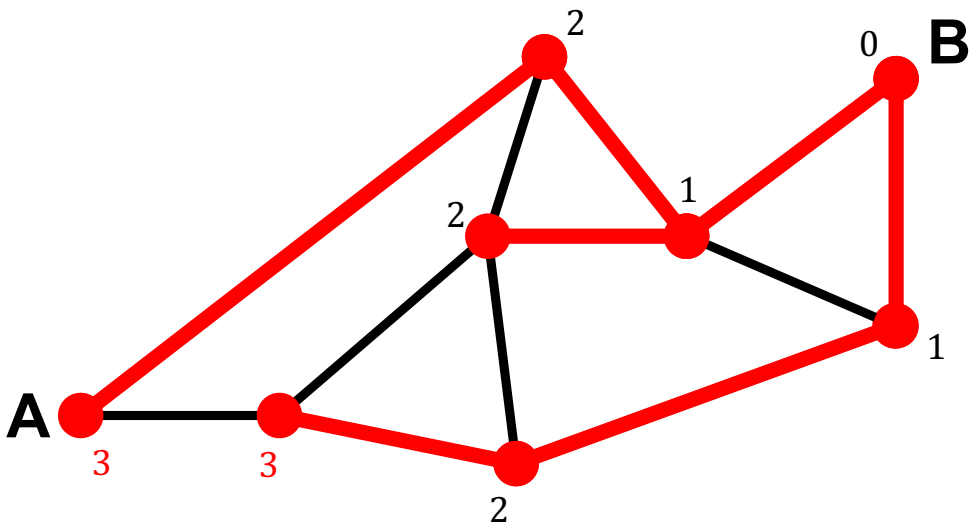
Algorithme BFS

1. Les **voisins directs** de B sont à distance 1 de celui-ci. On les marque également.
2. Les **voisins des voisins** sont à distance 2 de celui-ci. On les marque également.



Algorithme BFS

1. Les **voisins directs** de B sont à distance 1 de celui-ci. On les marque également.
2. Les **voisins des voisins** sont à distance 2 de celui-ci. On les marque également.
3. Et ainsi de suite... jusqu'à atteindre **tous les nœuds du graphe**, inclus le nœud A (ici, 3 étapes suffisent).



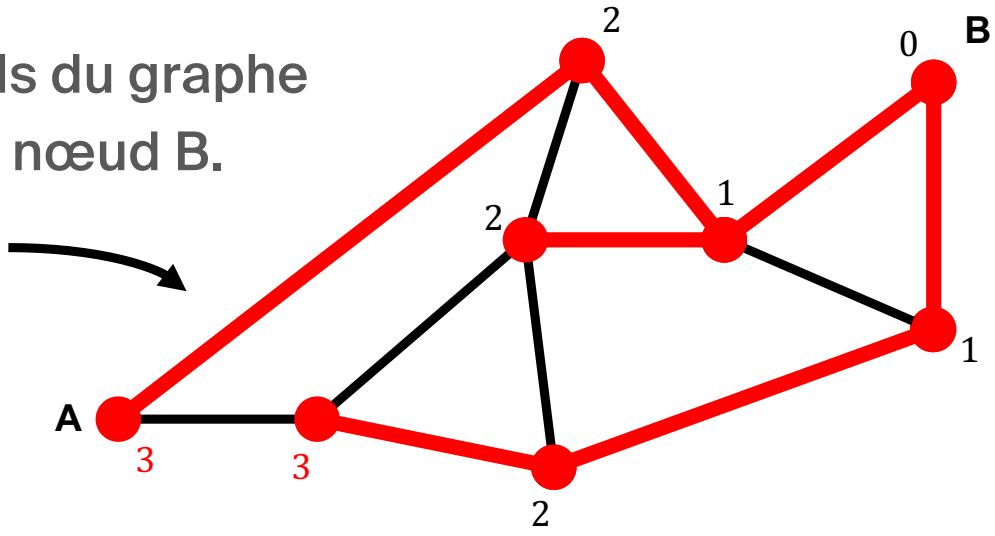
Algorithme BFS

0. Initialisation : le nœud B est à distance 0 de lui-même. On marque celui-ci comme vu.

1. Les **voisins directs** de B sont à distance 1 de celui-ci. On les marque également.
2. Les **voisins des voisins** sont à distance 2 de celui-ci. On les marque également.
3. Et ainsi de suite... jusqu'à atteindre **tous les nœuds du graphe**, inclus le nœud A.

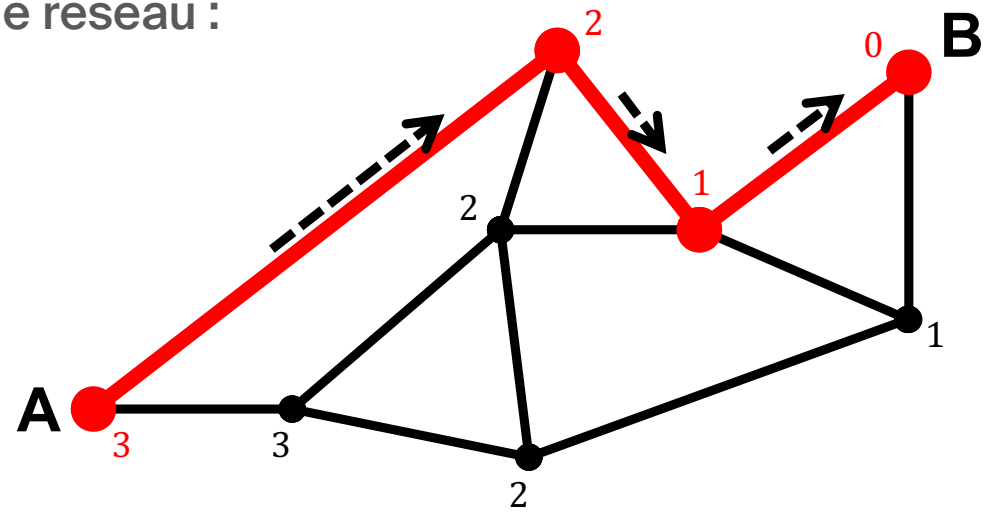
- À la fin de l'algorithme, tous les nœuds du graphe connaissent leur plus petite distance du nœud B.

arbre couvrant minimal !



Routage

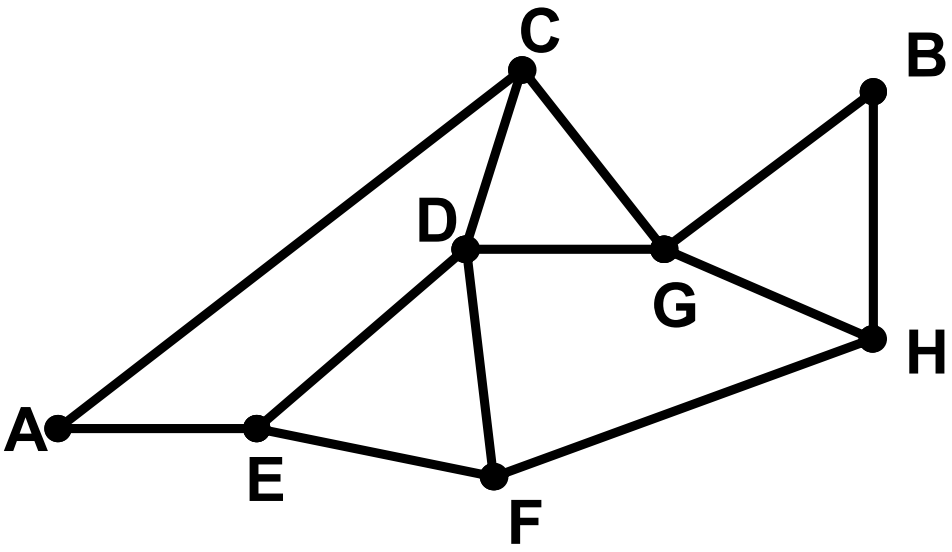
L'algorithme BFS nous donne maintenant une idée de comment acheminer les paquets à travers le réseau :



Supposons que A désire envoyer un paquet à B :
À quel voisin direct va-t-il choisir de transmettre le paquet ? À celui dont la distance à B est la plus petite ! (et ainsi de suite jusqu'à la destination)

Routage

Pour que cela fonctionne en pratique, il importe que chaque nœud du réseau maintienne à jour une **table de routage** contenant les informations suivantes :



Nœud A		
Destination	**Direction**	**Distance**
B	C	3
C	C	1
D	C ou E	2
E	E	1
...	...	...

Nœud C		
Destination	**Direction**	**Distance**
A	A	1
B	G	2
D	D	1
E	A ou D	2
...	...	...

Quelques remarques

- La mise à jour des tables de routage s'effectue **de manière distribuée**, chaque nœud vérifiant à intervalles réguliers les informations de ses voisins directs.

- En pratique, chaque nœud ne tient pas à jour une table avec toutes les destinations possibles, mais **seulement les destinations proches de lui** ; la gestion des destinations plus lointaines est déléguée à un nœud plus important (« gateway ») → **hiérarchie dans le réseau**

Adresses IP *(v4 sur 32 bits)*

Chaque nœud dans le réseau possède sa propre adresse IP, qui reflète la hiérarchie de celui-ci.

- Exemple :

$$172.16.254.1$$

(chaque nombre $\in \{0 \dots 255\}$)

8 bits 8 bits 8 bits 8 bits

$\rightarrow 32$ bits au total

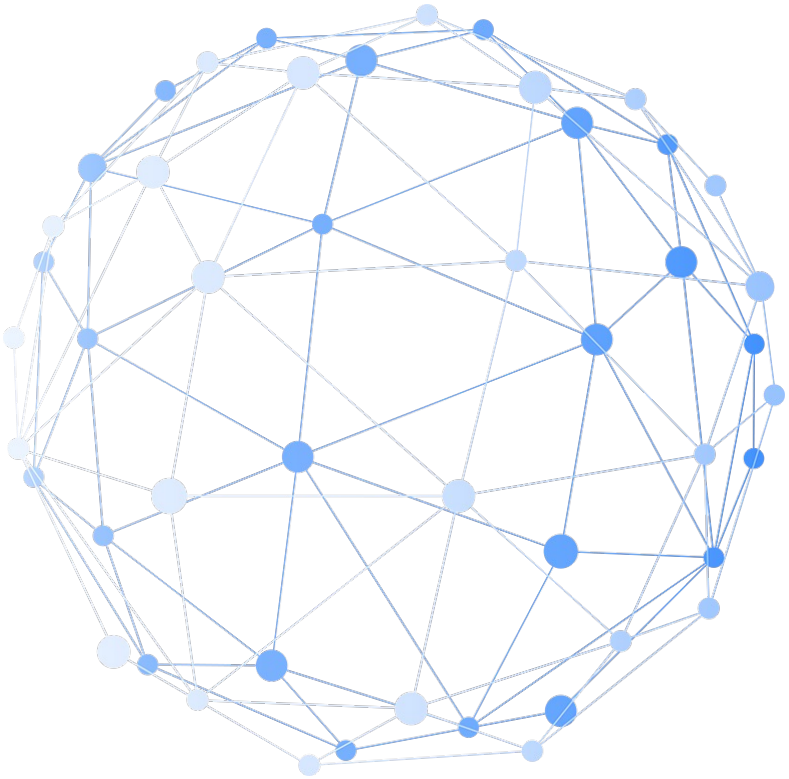
$\rightarrow 2^{32}$ adresses différentes

~ 4 milliards

réseau principal — sous-réseau — … — …

Mais pas assez en 2020 ! → IP v6, sur 128 bits (On espère assez pour quelque temps !)

Algorithme BFS (Breadth-First Search)



- Distance minimale depuis la destination
- Construction d'un arbre couvrant
- Permet à chaque nœud de connaître le meilleur voisin pour atteindre la cible
- Chaque machine stocke :
  - Destination
  - Voisin suivant (next hop)
  - Distance (nombre de sauts)
- Mise à jour par ouï-dire (périodique)

Aujourd'hui

- Introduction aux réseaux informatiques :
  paquets et couches de protocoles
- Routage (Protocole IP)
  - Comment les données trouvent leur chemin (routage/IP)

- **Introduction à la Cryptographie**
  - **Chiffrement symétrique**
  - Protocole de Diffie-Hellman
  - Chiffrement asymétrique

La sécurité informatique

- **Les besoins :**
  - Confidentialité
  - Intégrité
  - Authentification
  - Non-répudiation
- **Les briques :**
  - Chiffrement
  - Échange de clés
  - Hachage
  - Signatures

Les besoins

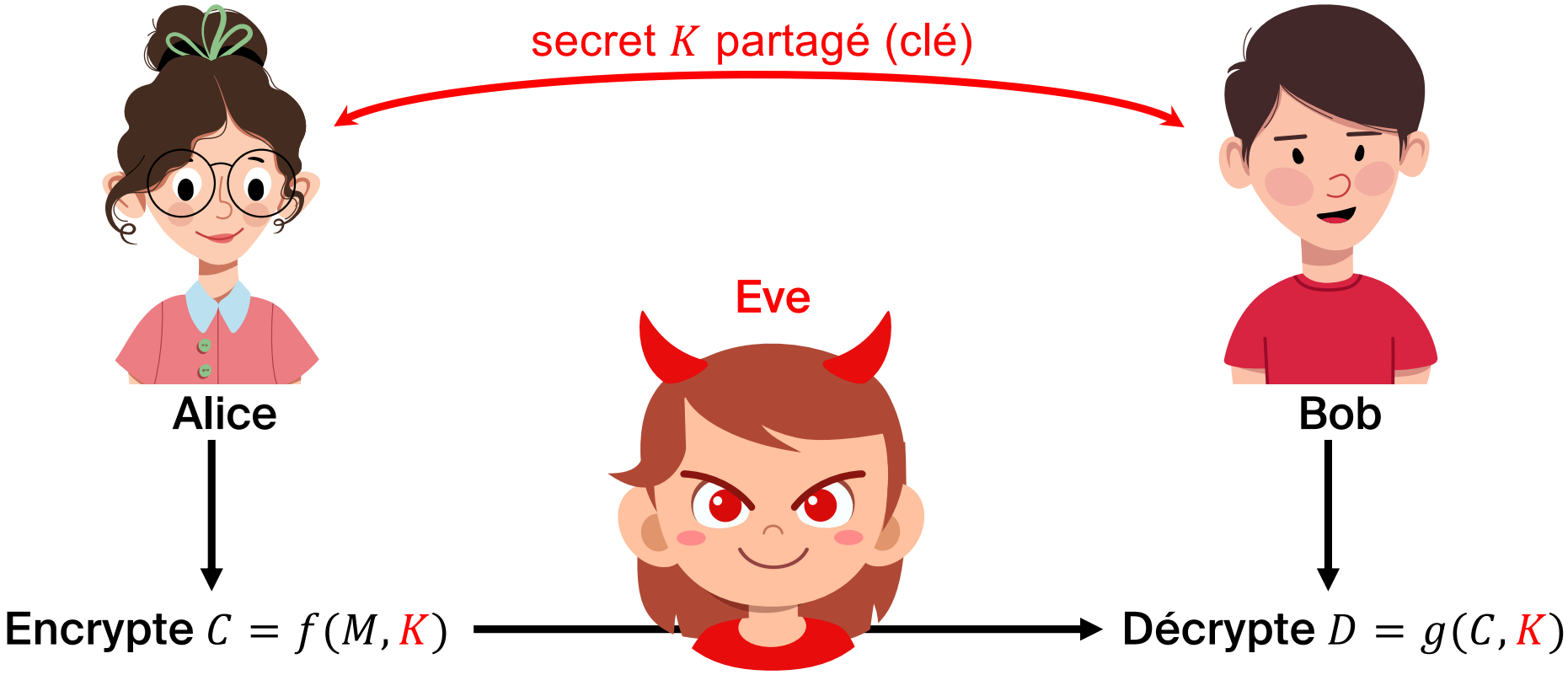
- **Confidentialité :** seul le destinataire peut lire
  - Réalisée via chiffrement
  - Empêche qu'un message intercepté soit lisible par un tiers non autorisé
- **Intégrité :** le message n'a pas été modifié
  - Détecte toute altération, même minime, du contenu
  - Utilisation typique des fonctions de hachage
  - Éviter les falsifications silencieuses

Les besoins

- **Authenticité :** s'assurer de l'identité de l'expéditeur
  - Utilise des signatures numériques ou des mécanismes d'authentification (mot de passe, biométrie)
  - Évite les attaques par usurpation d'identité

- **Non-répudiation :** L'émetteur ne peut pas nier avoir envoyé le message
  - Repose sur les signatures numériques et l'usage de clés privées
  - Important dans les contextes juridiques ou contractuels

Chiffrement symétrique : scénario



Eve intercepte C mais ne sait pas trop quoi en faire sans la clé K...

Chiffrement de César

- Chiffrement par décalage de lettres dans l'alphabet
- Exemple avec un décalage de 3 :
  - Message : BONJOUR
  - Clé : +3
  - Chiffré : ERQMRXU
- **Facile à casser** : il n'y a que 25 possibilités
- Sert surtout à illustrer le principe de substitution

Clé à usage unique *(« one-time pad »)*

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
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Message M	
C	3
O	15
U	21
C	3
O	15
U	21

+

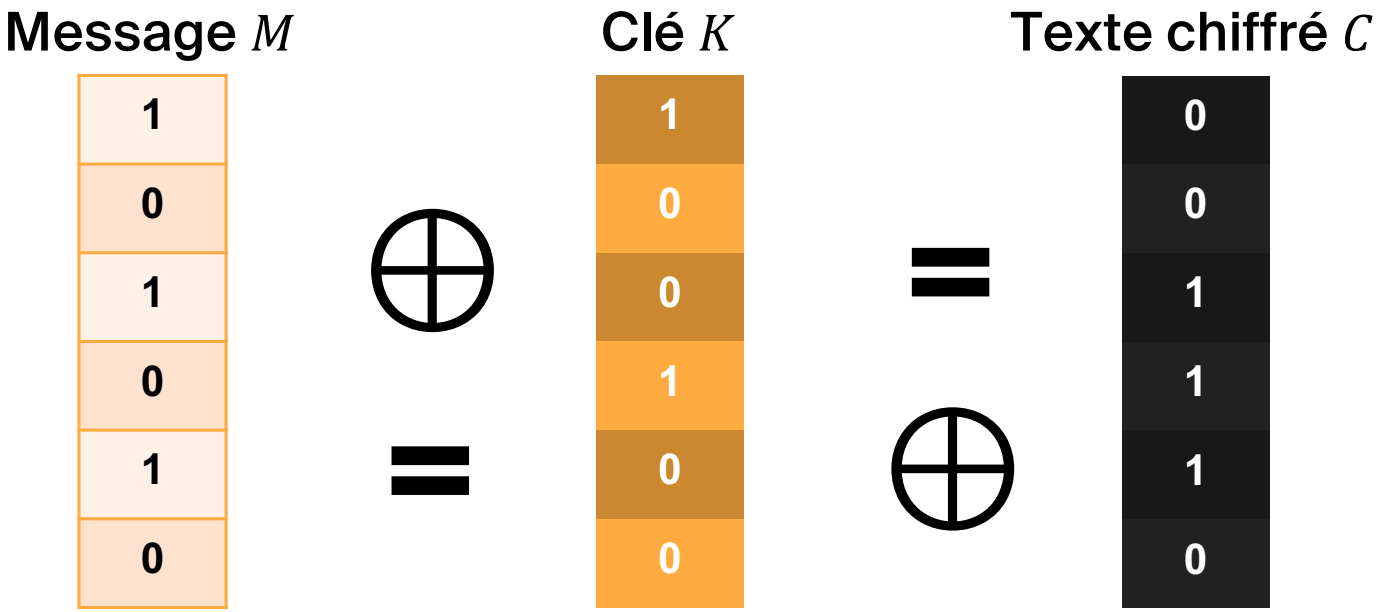
Clé K	
S	19
E	5
C	3
R	18
E	5
T	20

=

Texte chiffré C	
V	22
T	20
X	24
U	21
T	20
O	15

$$C = (M + K) \bmod n \longrightarrow D = (C - K) \bmod n$$

Clé à usage unique *(« one-time pad »)*



$C = M \oplus K$ ⟶ $D = C \oplus K$

$D = M \oplus K \oplus K$

$D = M$

Pas de débordement 👍

Clé à usage unique *(« one-time pad »)*

	Addition modulaire	XOR
Domaine	Entiers (e.g., 8, 16, 32 ou 64 bits)	Données binaires (bit par bit)
Chiffrement	$C = (M + K) \bmod n$	$C = M \oplus K$
Déchiffrement	$D = (C - K) \bmod n$	$D = C \oplus K$
Remarque	Nécessite une soustraction pour déchiffrer	Plus rapide, auto-inverse

Clé à usage unique *(« one-time pad »)*

- Le OTP assure une sécurité parfaite
  à condition que la clé :
  - soit parfaitement **aléatoire**
  - ait exactement la **même taille** que le message
  - ne soit **jamais réutilisée**

- Le One-Time Pad ne repose pas sur un problème mathématique difficile, mais sur une
  clé parfaitement aléatoire et unique.
  - Cela le rend **résistant aux attaques quantiques**.

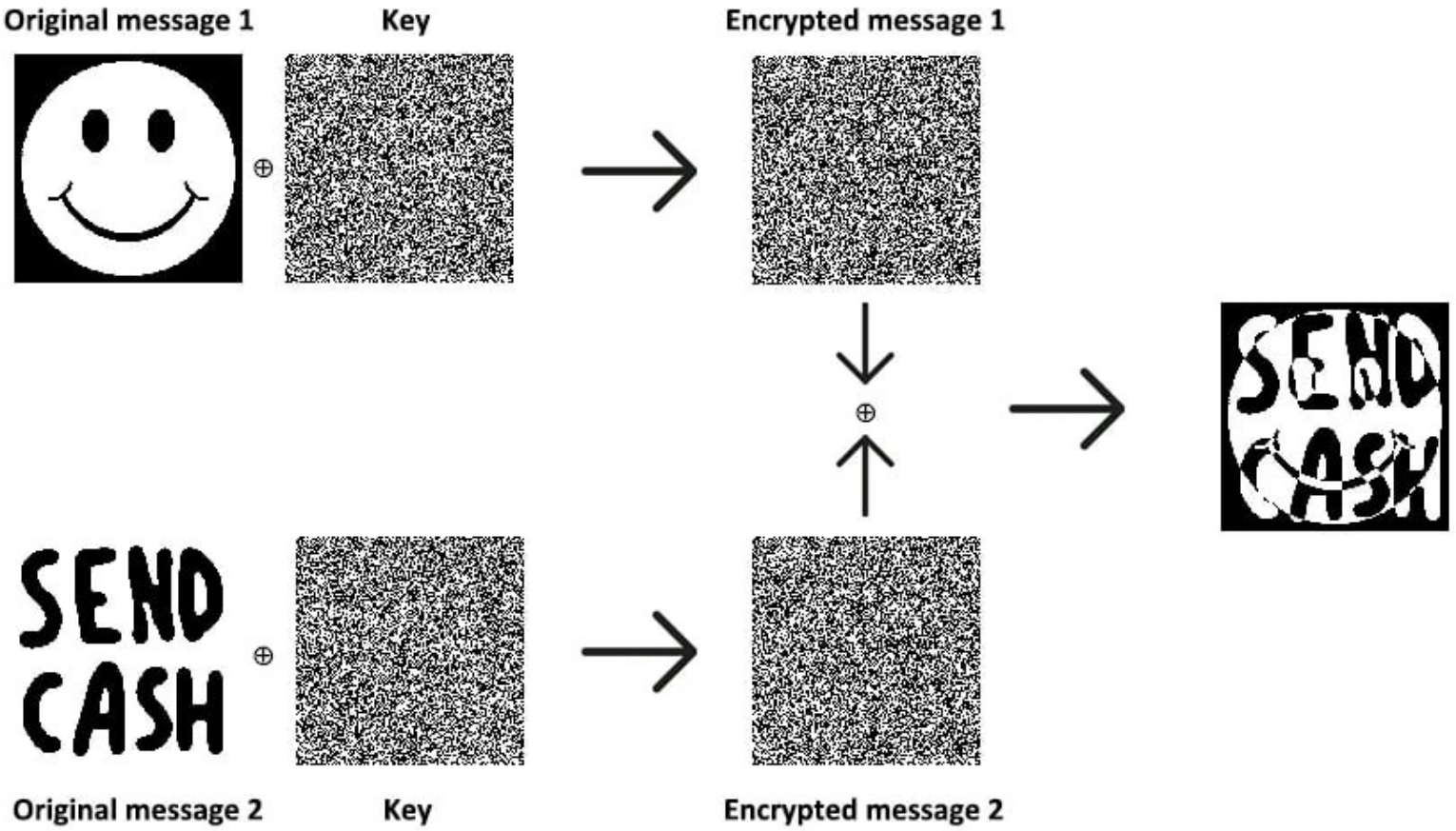
Défauts du OTP

- Pour envoyer un message de longueur n,
  il faut une **clé aussi longue que le message**
- Cette clé doit être **transmise de façon secrète**
  avant toute communication
- Si la clé est trop simple (ex. $K = 00000000$),
  la sécurité est compromise
- Si la même clé est utilisée deux fois :

$$C1 = M1 \oplus K$$
$$C2 = M2 \oplus K$$
$$C1 \oplus C2 = M1 \oplus K \oplus M2 \oplus K = M1 \oplus M2$$

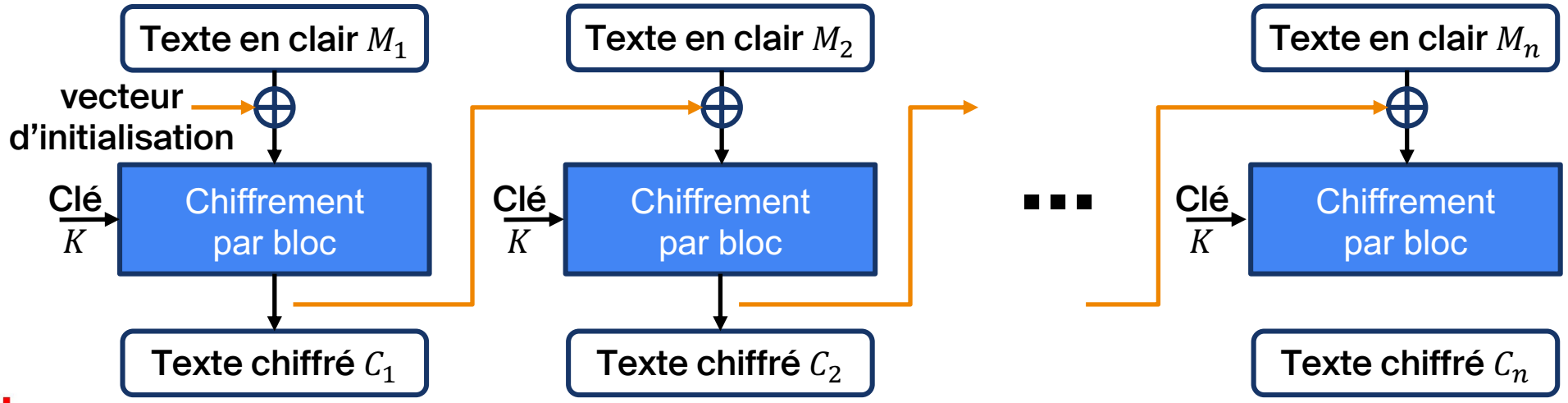
- Eve peut analyser les relations entre $M1$ et $M2$:
  plus aucune sécurité !

Défauts du OTP



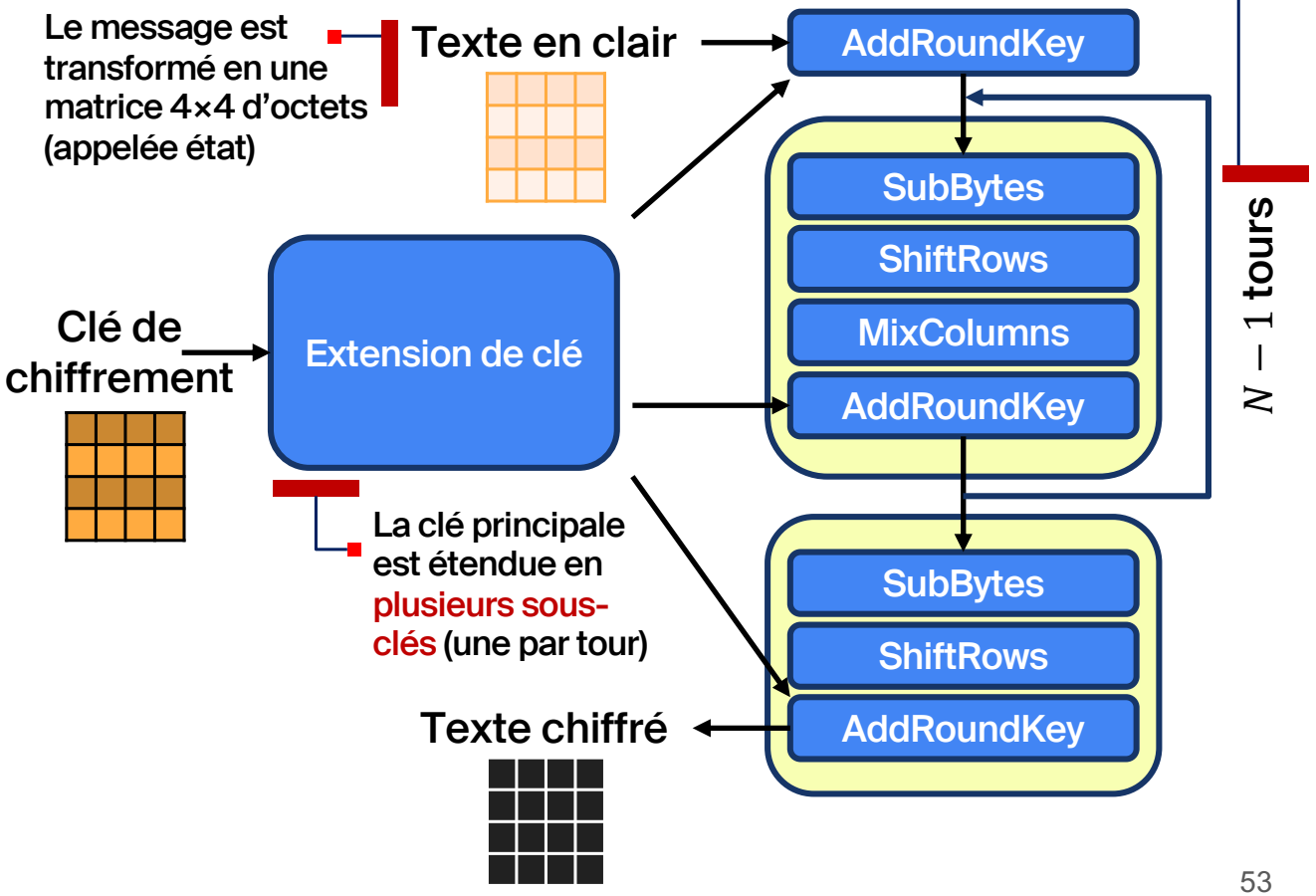
Chiffrements par blocs

- Contrairement au One-Time Pad, on utilise souvent **la même clé plusieurs fois**
- Un block cipher est un algorithme de chiffrement symétrique qui :
  - Prend un bloc de données M_i de taille fixe (par exemple 128 bits)
  - Utilise une clé secrète K
  - Et renvoie un bloc chiffré C_i de la même taille



AES : Advanced Encryption Standard

- Bloc de 128 bits
- Clés de 128, 192 ou 256 bits
- Fonctionne en 10, 12 ou 14 tours
- Chaque tour applique : substitution, permutation, mélange, et ajout de sous-clé
- Utilisé partout : Wi-fi, HTTPS, VPN, disques chiffrés



Bilan : Chiffrement symétrique

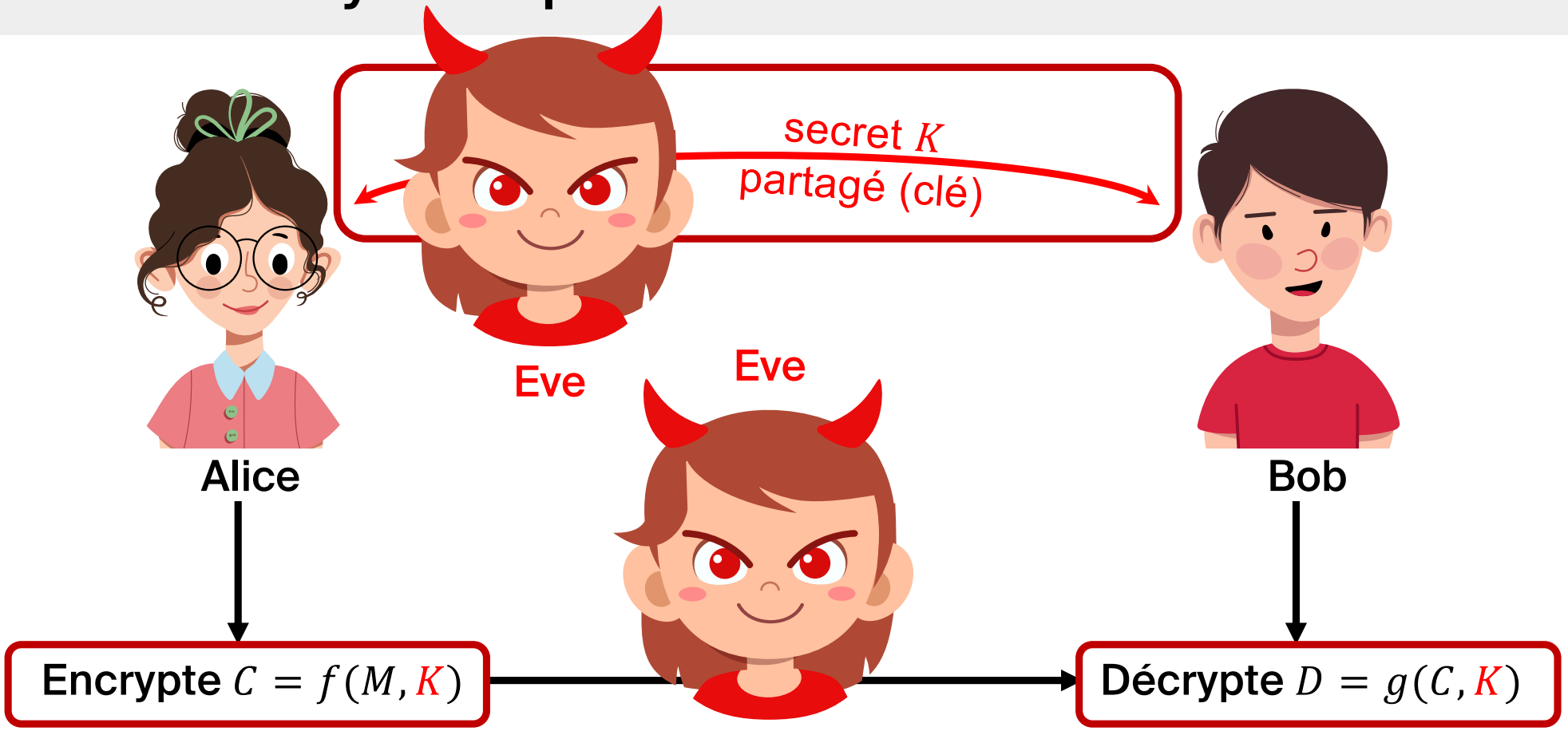
- Une seule clé pour chiffrer / déchiffrer
- Exemples : César, OTP, AES
- La taille du message chiffré est **la même** que celle du message d'origine
- Problème : **comment partager la clé sans la divulguer ?**

Aujourd'hui

- Introduction aux réseaux informatiques :
  paquets et couches de protocoles
- Routage (Protocole IP)
  - Comment les données trouvent leur chemin (routage/IP)

- Introduction à la Cryptographie
  - Chiffrement symétrique
  - **Protocole de Diffie-Hellman**
  - **Chiffrement asymétrique**

Chiffrement symétrique : scénario



Eve intercepte C mais ne sait pas trop quoi en faire sans la clé K...

Protocole Diffie-Hellman : version simplifiée

- Alice et Bob désirent échanger des informations de manière confidentielle **sans** disposer d'une clé secrète K échangée au préalable.

Supposons qu'Eve a raté l'école et n'a jamais appris à diviser. En revanche, elle est super douée pour intercepter toutes les communications entre Alice et Bob. Ils ne peuvent pas échanger un mot sans qu'Eve n'en soit à l'écoute.

Protocole Diffie-Hellman : version simplifiée

On a donc besoin d'**opérations difficilement inversibles** ou « **opérations à sens unique** ».

Même si Eve intercepte M, N_{AM} et N_{BM}, **tant qu'elle ne sait pas diviser**, le secret reste bien protégé.

1. Ils se mettent d'accord sur un 3ème nombre public M

M

1. Alice choisit un nombre secret N_A
2. Alice calcule $N_{AM} = N_A \cdot M$
3. Alice envoie N_{AM} à Bob
4. Alice calcule $N_A \cdot N_{BM} = \boxed{N_A \cdot N_B \cdot M} = K$

N_{AM}

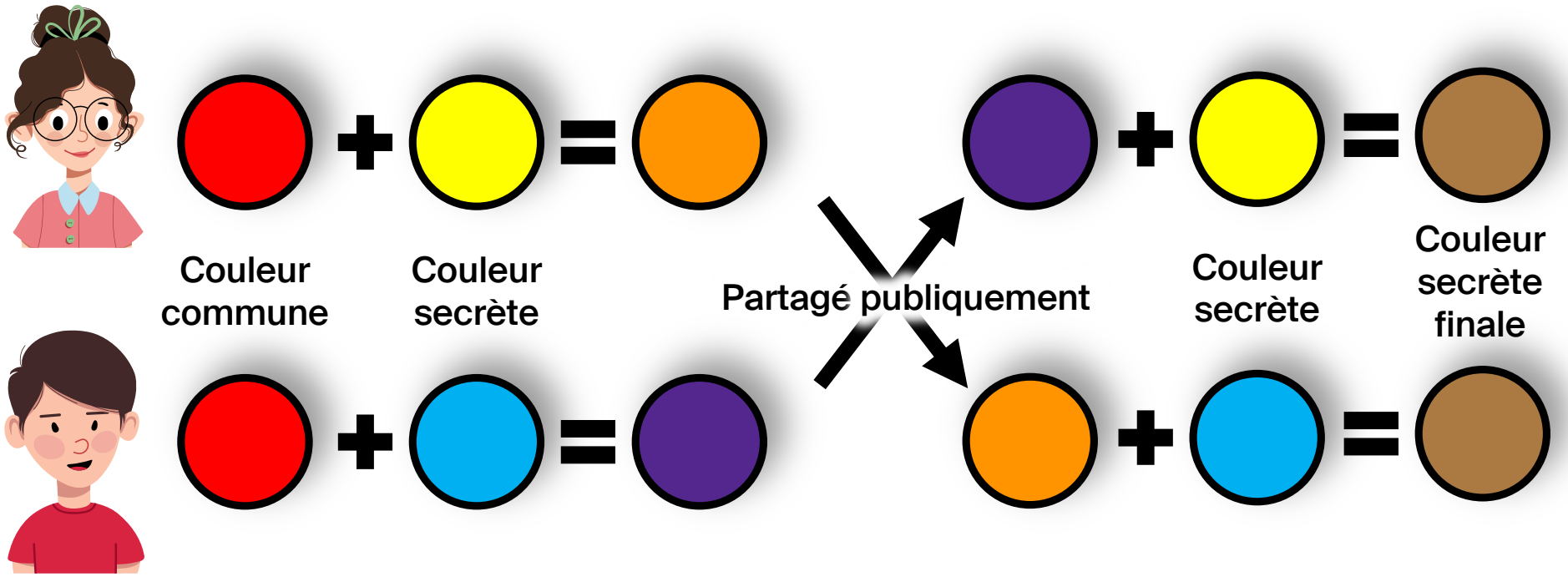
N_{BM}

1. Bob choisit un nombre secret N_B
2. Bob calcule $N_{BM} = N_B \cdot M$
3. Bob envoie N_{BM} à Alice
4. Bob calcule $N_B \cdot N_{AM} = \boxed{N_A \cdot N_B \cdot M} = K$

égaux = secret partagé

Diffie-Hellman avec des couleurs

- Les couleurs sont facilement mélangeables, mais difficilement séparables



Arithmétique modulaire

- **Soit P un grand nombre premier. Sur l'ensemble $\{0,1, \dots, P-1\}$, on définit l'addition, la multiplication et l'exponentiation modulo P :**

Exemples (avec $P = 5$) :

$a + b \pmod P$	$a \cdot b \pmod P$	$a^b \pmod P$
$4 + 3 \ (mod\ 5) = 2$	$4 \cdot 3 \ (mod\ 5) = 2$	$4^3 \ (mod\ 5) = 4$

- pas de dépassement de capacité
- toutes des opérations faciles à exécuter

Arithmétique modulaire

$a^b \pmod P$
$4^3 \ (mod\ 5) = 4$

- Il se trouve que l'opération $a^b (mod\ P) = c$ est **difficile à inverser** (i.e., si on nous donne P, a et c, **il est difficile de retrouver** b).

Voilà donc **l'opération à sens unique** que nous allons utiliser.

Remarques

- La difficulté de l'opération d'inversion dépend des nombres P et a choisis (c'est le problème du **logarithme discret**).
- Il existe par contre des **algorithmes efficaces** pour trouver de **grands nombres premiers** et donc réaliser concrètement ce qui va suivre !

Protocole Diffie-Hellman

Même si Eve intercepte P, Q, N_{QA} et N_{QB}, tant qu'elle ne sait pas résoudre le problème du logarithme discret en un temps raisonnable, le secret reste bien protégé.

1. Alice et Bob choisissent ensemble un grand nombre premier P, ainsi qu'un autre nombre Q entre 1 et $P-1$.

P, Q

1. Alice choisit un nombre secret $1 < N_A < P-1$
2. Alice calcule $N_{QA} = Q^{N_A} \pmod P$
3. Alice envoie N_{QA} à Bob
4. Alice calcule ${N_{QB}}^{N_A} = Q^{N_A \cdot N_B} \pmod P = K$

N_{QA}

N_{QB}

1. Bob choisit un nombre secret $1 < N_B < P-1$
2. Bob calcule $N_{QB} = Q^{N_B} \pmod P$
3. Bob envoie N_{QB} à Alice
4. Bob calcule ${N_{QA}}^{N_B} = Q^{N_A \cdot N_B} \pmod P = K$

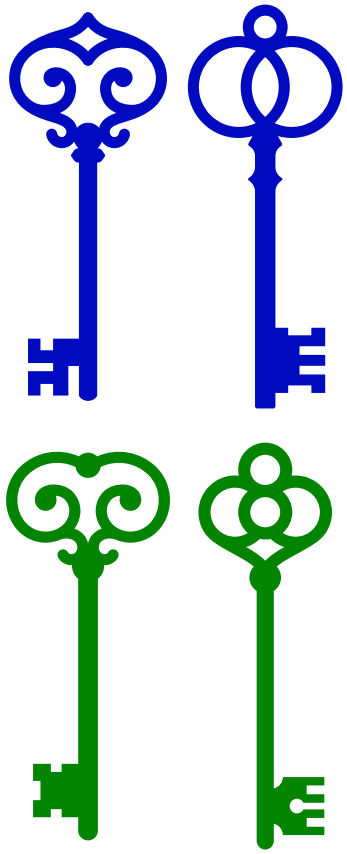
égaux = secret partagé

Aujourd'hui

- Introduction aux réseaux informatiques :
  paquets et couches de protocoles
- Routage (Protocole IP)
  - Comment les données trouvent leur chemin (routage/IP)

- Introduction à la Cryptographie
  - Chiffrement symétrique
  - Protocole de Diffie-Hellman
  - **Chiffrement asymétrique**

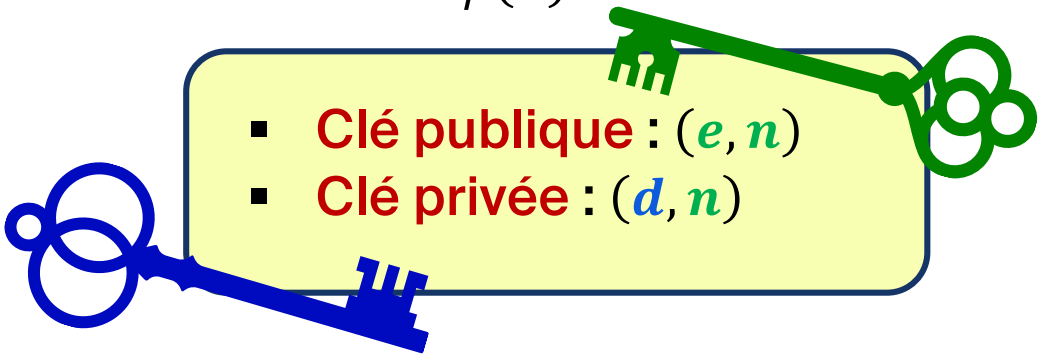
Cryptographie asymétrique



- Chaque utilisateur possède deux clés :
  - Une clé publique (diffusée librement)
  - Une clé privée (gardée secrète)
- Utilisations typiques :
  - Chiffrement → avec la clé publique du destinataire
  - Signature → avec sa propre clé privée
- Avantage : plus besoin de partager un secret à l'avance
- Inconvénient : calculs plus lourds que les méthodes symétriques
- La sécurité repose sur :
  - Multiplier deux grands nombres premiers : facile
  - Retrouver les facteurs à partir du produit : très difficile (en pratique)

Cryptographie asymétrique : RSA

- Alice choisit deux grands nombres premiers p et q
- Elle calcule :
  - $n = p \cdot q$
  - $\varphi(n) = (p-1) \cdot (q-1)$
- Elle choisit un exposant public e, tel que :
  - $1 < e < \varphi(n)$, et $\gcd(e, \varphi(n)) = 1$
- Elle calcule ensuite l'exposant privé d, tel que :
  - $e \cdot d \equiv 1 \bmod \varphi(n)$

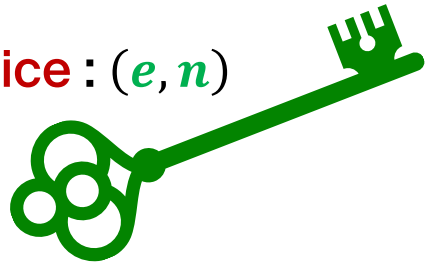


Cryptographie asymétrique : RSA

- Bob veut envoyer un message secret à Alice

- Il utilise la clé publique d'Alice : (e, n)
- Il chiffre son message m :

$$c = m^e \bmod n$$



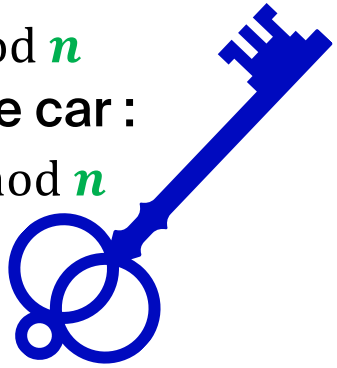
c

- Alice reçoit c et utilise sa clé privée (d, n) pour déchiffrer :

$$m = c^d \bmod n$$

- Cela fonctionne car :

$$m^{e \cdot d} \equiv m \bmod n$$



- En pratique, on ne chiffre pas directement un message long avec RSA
- On chiffre une clé de session → utilisée ensuite avec AES

Signature numérique

- On ne chiffre pas le message, mais son empreinte (hash) **avec sa clé privée**
- Tout le monde peut **vérifier** avec la clé publique

- Pourquoi c'est utile ?
  - Garantit **l'authenticité** : seul le propriétaire de la clé privée peut signer
  - Garantit **l'intégrité** : toute modification du message entraîne un hash différent
  - Garantit **la non-répudiation** : la signature prouve que l'auteur est bien à l'origine du message

Fonctions de hachage cryptographiques

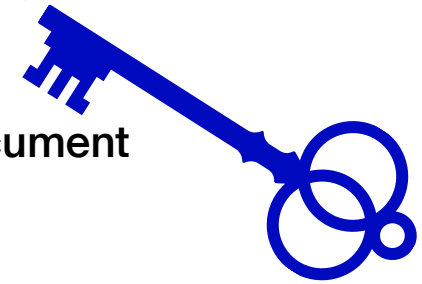
- Une fonction de hachage prend n'importe quelle donnée
  (texte, vidéos, musique…)
  et produit une empreinte numérique courte (ex. 256 bits pour SHA-256)

- Propriétés essentielles :
  - Déterministe : même entrée → même sortie
  - Sens unique : impossible de retrouver l'entrée à partir du hash
  - Résistance aux collisions : très difficile de trouver deux entrées différentes ayant le même hash

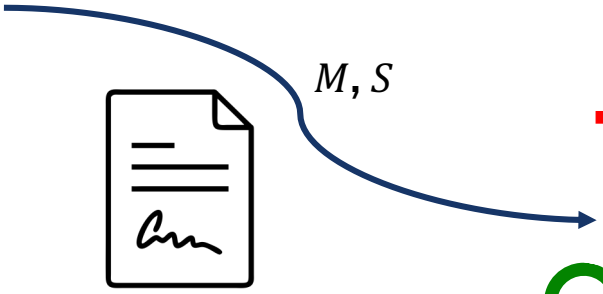
- Exemples : SHA-256, SHA-3, MD5 👎, SHA-1 👎

Signature numérique

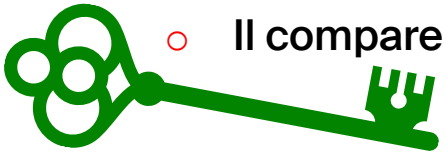
- Alice veut envoyer un document signé à Bob



- Alice calcule $\texttt{hash}(M)$ où M est le document
- Elle chiffre ce hash avec sa clé privée.
  Cela produit la signature S
- Elle envoie à Bob le document M et la signature S



- Bob vérifie la signature :
  - Il calcule $\texttt{hash}(M)$ de son côté
  - Il déchiffre S avec la clé publique d'Alice
  - Il compare les deux résultats



Aujourd'hui

- Introduction aux réseaux informatiques :
  paquets et couches de protocoles
- Routage (Protocole IP)
  - Comment les données trouvent leur chemin (routage/IP)

- Introduction à la Cryptographie
  - Chiffrement symétrique
  - Protocole de Diffie-Hellman
  - Chiffrement asymétrique

Résumé Semaine 3 – Réseaux – ICC-T

- Un **réseau** permet à des machines de communiquer par **échange de paquets**
- La communication suit une **pile de couches** (TCP/IP) pour structurer les échanges
- Chaque couche a un **rôle spécifique** et ajoute (ou retire) un **en-tête**
- Les **adresses** changent selon la couche : MAC (locale), IP (globale), ports (processus)
- Le **routage** permet de trouver le chemin à travers le réseau (IP + tables)

Résumé Semaine 3 – Crypto – ICC-T

- Les besoins en sécurité : confidentialité, authenticité, intégrité, non-répudiation
- Les outils cryptographiques :
  - Chiffrement symétrique (AES) : rapide, clé partagée
  - Diffie-Hellman : permet un échange de clé sans la transmettre
  - Chiffrement asymétrique (RSA) : deux clés, mais plus lent
  - Signature numérique : prouve l'identité et l'intégrité d'un message
- Certains cryptosystèmes (ex. : RSA, Diffie–Hellman) reposent sur des problèmes mathématiques difficiles (ex. : factorisation, logarithme discret) ; d'autres (AES, OTP) s'appuient sur des constructions algorithmiques ou de l'aléa parfait.

rafael.pires@epfl.ch

Merci