Algo q2

Créateur	Robin Gillard (el_pablo__)

Algorithme : méthode qui permet de résoudre un problème de façon systématique

▼ Rappel

Structure de controle

- Séquences
- Conditionnelles
- Boucles

Structure de données

- Constantes
- variables
- Tableaux
- Structures récursives

Les preuves

- Terminaison : s’assurer que l’algorithme terminera en un temps fini
- Connection : s’assurer que le résultat fourni par l’algorithme est la solution au problème
- Complétude : s’assurer que l’algorithme donne une bonne réponse peu importe la variable

Revoir les tris

Revoir les tableaux et listes chainées

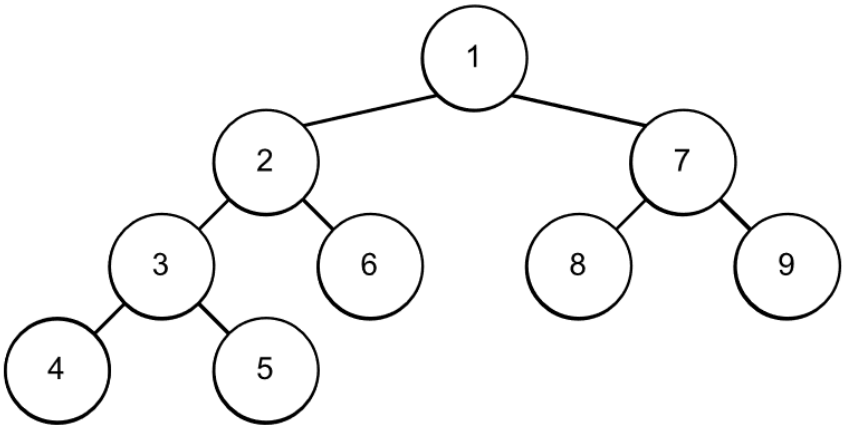
Les arbres

- Le premier éléments de l'arbre serra toujours la racine
    - Racine : Le début de l'arbre, c'est "l'origine"
    - Nœud (Node) : élément d'un arbre qui comprend une valeur ainsi qu'une référence vers d'autres nœud
    - Feuille (Leaf) : élément en bout d'arbre qui n'a pas d'enfants
    - Enfants : référence d'un Nœud dit parent
    - Parent : Nœud qui faire références a des Nœud dits enfants
    - Frères : Nœud qui ont le même parents
    - Profondeur (Niveau): Distance entre le nœud et la racine
- Hauteur : profondeur maximale d'un nœud
- Représentation hiérarchique des données
- Utilisation régulière

Les arbres binaires

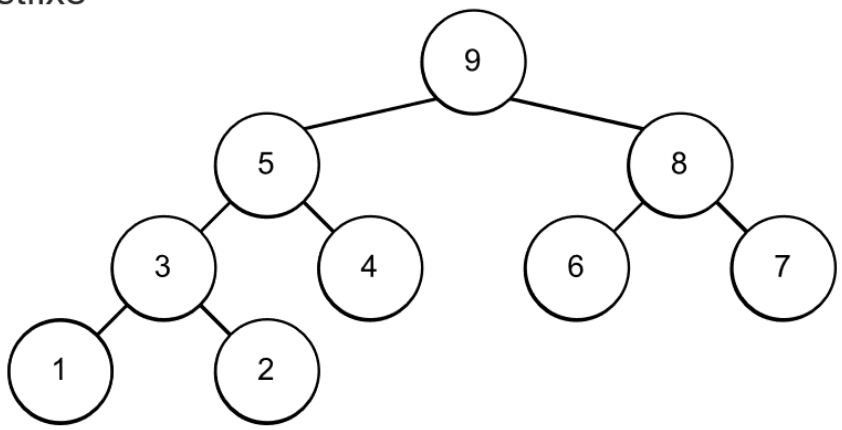
- Un arbre dont chaque parents ne peuvent avoir au maximum que 2 enfants
- Différentes méthodes de parcours
    - En profondeur
        - Préfixe
            - On commence par les parents, aller au plus au fond possible puis quand on peut plus avancer je recule d'1 et regarde ceux que j'ai pas encore regardé
            -

- Préfixe



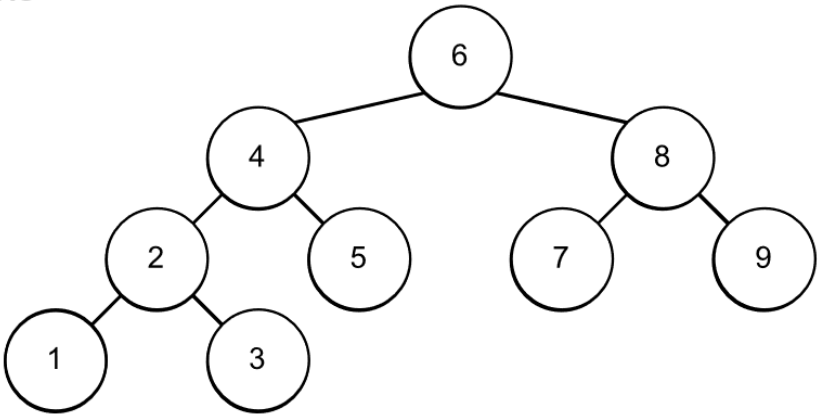
- Postfixe
  - On commence par les enfants
  -

- Postfixe



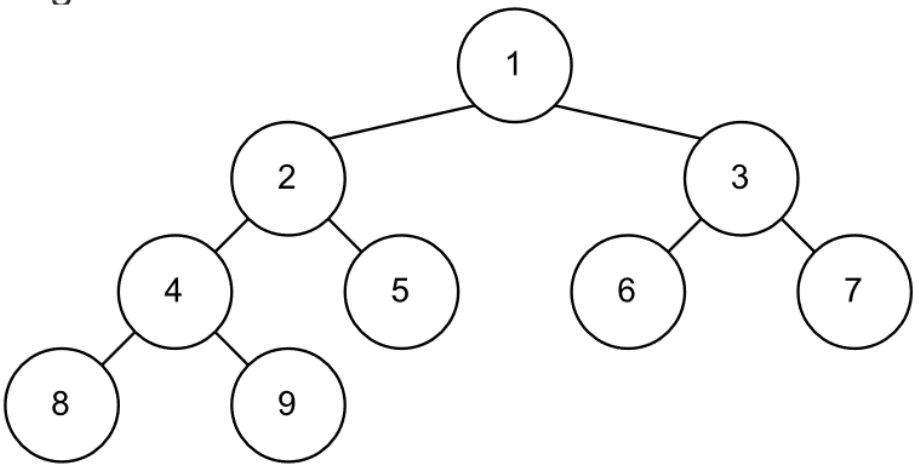
- Infixe
  -

- Infixe



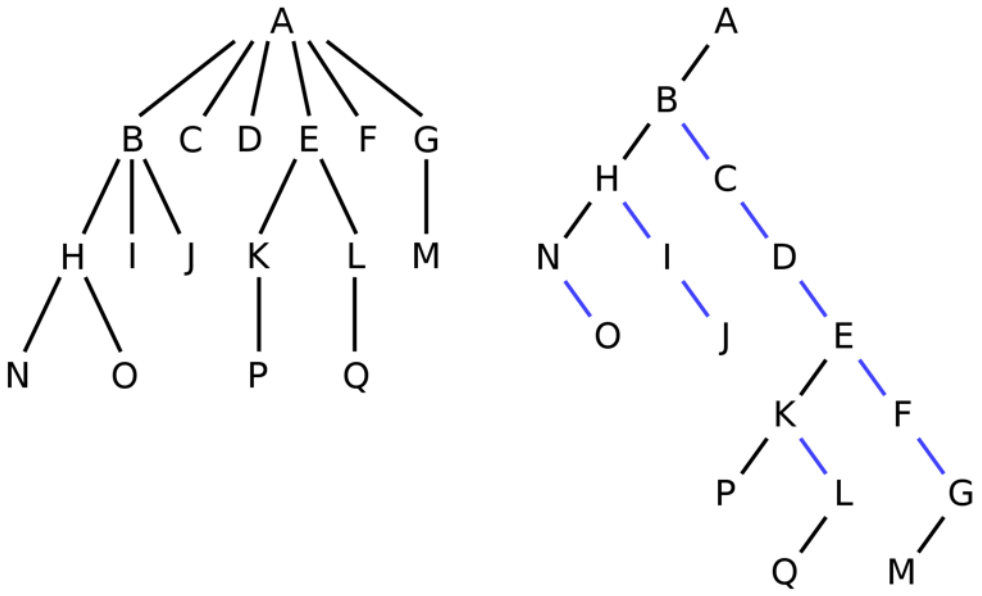
  - En largeur
    - Traiter les 2 nœuds en même temps de gauche a droite
    -

- En largeur



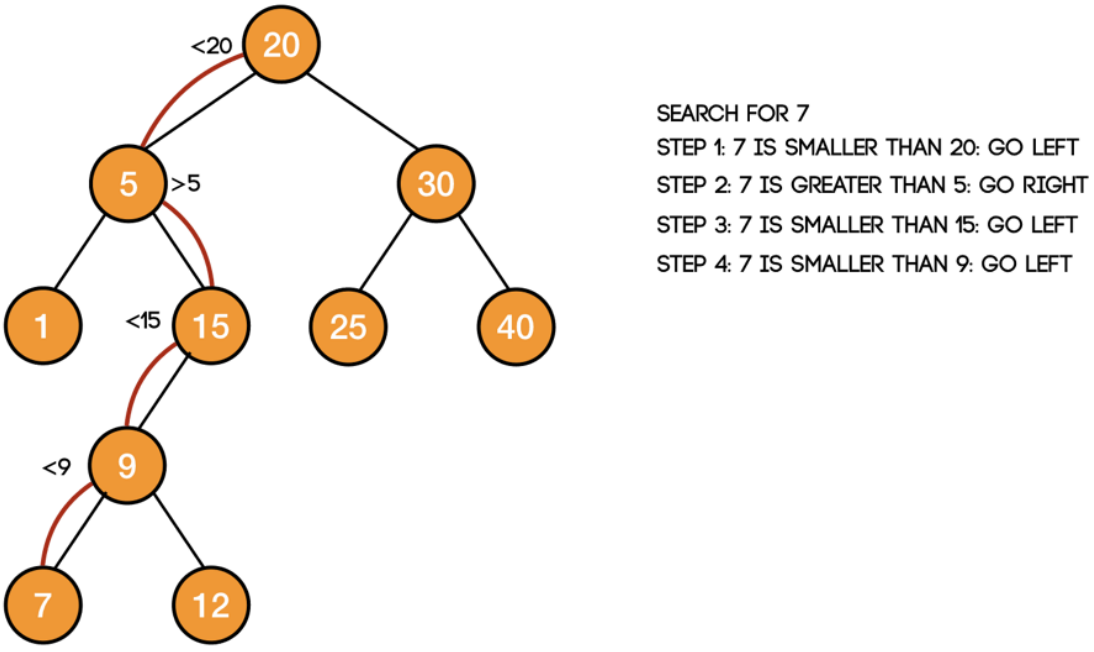
- Il est possible de transformer un arbre n-aire en arbre binaire en mettant en enfants les frères (vers la droite) et en enfants les enfants déjà présent (vers la gauche)

- Transformer un arbre *n*-aire en arbre binaire



Arbres binaire de recherche

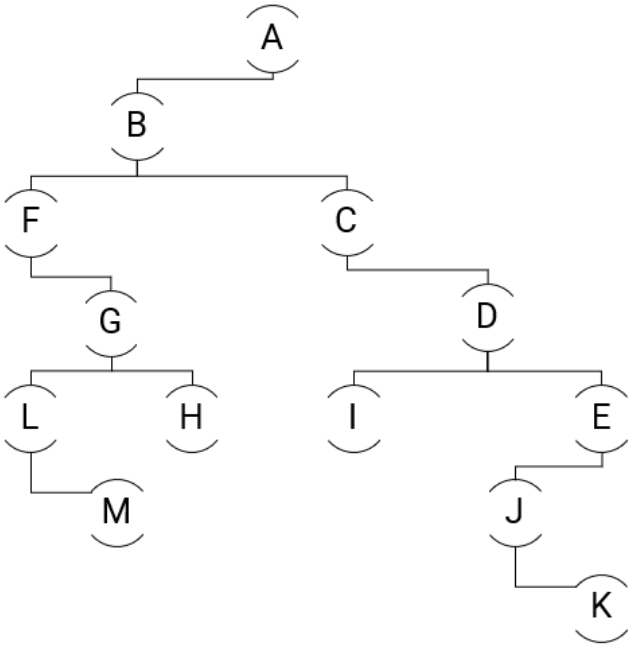
- Arbre dans lequel chaque nœuds possède une clé (valeur) de tel sorte que chaque nœuds du sous-arbre de gauche ait une clé plus petite ou égale au nœuds considéré et plus grand pour le sous-arbre droit



- Si l'arbre est vide retourner nœuds non trouvé

- Si la valeur cherchée est égale a la valeur du nœuds courant retourner le nœuds courant
- Si la valeur cherchée est inférieure à la valeur du nœuds courant, rechercher dans le sous arbre gauche
- Complexité temporelle
  - Pire des cas 0(n)
  - Dans les autres cas, tout dépend de la structure de l'arbre
    - Arbre équilibré : 0log(n)
    - Arbre pas équilibré 0(n)
- Insertion
  - Chercher pour sa place au niveau d'un nœuds et l'ajouter en tant que fils de ce nœuds
- suppression
  - Quand on veut supprimer un parents avec pas mal d'enfants, le remplacer par successeur (le plus petit des plus grand) ou par son prédécesseur (plus grand des plus petit)

▼ **EXERCICE PREFIXE ETC**



- Préfixe
  ABFGLMHCDIEJK
- Postfixe
  MLHGFIKJEDCBA
- Infixe
  FLMGHBCIDJKEA

```
En vrai, la paketa dans le cul c pas une mauvaise idée ptn
```

▼ Rappel

≠ type de parcours :

- Parcours
- Profondeur
  - Préfixe
  - postfixe
  - infixe

Arbre de recherche : arbre binaire dans lequel chaque noeud possede une clé de tel sorte que chaque noeud du sous arbre ...

Opération :

- Recherches
- Insertion
- Suppresion

▼ **Tri par tas**

Utilise les listes et les arbres

se base sur le fonctionnement de la structure de données du tas

- Un tas c'est :
  - Arbre binaire parfait
  - Respecte la propriété du tas : La clé d'un nœud parent a une plus haute priorité que les clés de ses enfants (max-heap)
- Se fait en associant une liste et un arbre binaire
- **VOIR LA PHOTO POWERPOINT P8-9**

Algorithme :

1. S'assurer qu'on respecte la propriété (max-heap)
2. Echanger la racine avec le dernier élément (swap)
3. Réduire la taille du tas de 1 (remove)
4. Repositionner l'élément le plus grand à la racine (heapify)
5. recommencer point 2

Utilité :

- Dans les OS
- En GUI
- Moteurs de recherches

▼ Les tables de hachage

Structure de données reposant sur des tableaux

- Structure de données statique
- Algorithme de recherche performant
- Comment ?
  - La position de l'élément dans le tableau est fonction de l'éléments lui-même
- Notion de clé :

    - Partie d’un élément qui permet de désigner le contenu de cet élément sans ambiguïté
    - Exemple :
        - Étudiant - Matricule
        - Abonné - Numéro de client
        - etc

Qu’est ce que le Hachage ?

- Comment stocke t-on les mots de passe ?
    - très souvent enregistrés mais illisible
        - Utilisation d’une fonction de hachage qui va calculer une empreinte unique à partir de données fournies
        - Attention données non récupérables
    - Propriétés
        - Longueur de la signature identique quelque soit la taille des données
        - Unidirectionnel
        - Impossible de prédire la signature
        - 2 données différentes = 2 signatures différentes
- Salage
    - Ajouter un élément a la donnée pour en modifier la signature
    - Augmente la sécurité face à différentes attaques
    - Eviter des doublons de mot de passes

exemple de fonctions :

- MD5 : peu sécurité et collision possible
- SHA-1 : moins collisions mais encore un peu
- SHA-2 : standards actuel

Format table hachage :

- Clé (John smith)
- Index
- Valeur clé (John Smith; 0476 54 87 96)

La recherche par clé est très simple : Hash de la clé = index dans le tableau

Sauf si collision

Solution collision :

- par chaînage **HELP ME**
- Port d'adressage ouvert

Complexité temporelle extrêmement intéressante pour chercher, ajouter ou supprimer un élément

Base de données, caches, etc

▼ SHA-256

1. Prétraitement des données
    a. Objectif : compatibilité des données avec l'algorithme
    b. En faisant du Padding
    c. Data lenght
    d. Permet de garantir l'intégrité des données (sa taille)
2. Initialisations des constantes
    a. Constantes (h0-h7) calculées a partir des 32 premiers bits et des 8 premiers nombres premiers
3. Division en bloc
4. Traitement des blocs
    a. Objectif : produire le hachage

b. Traitement récursif - traitement individuel

c. Transformation des blocs (permutation, rotation, modulo, etc..)

▼ Rappel

Savoir faire tout les types de tri

▼ Pathfinding

- Trouver le chemin le plus court entre un point A et un point B
- Différents critères :
  - La distance
  - Le coût
  - La vitesse

Approche intuitive

1. Prendre l'élément de fin et définir ses coordonnées ainsi que l'initialisation
   d'un compteur ici A = (1, 6, 0)
2. Ajouter cet élément à une liste sous forme FIFO
3. Parcourir la liste, en incluant les futurs éléments rajoutés et réaliser les
   opérations suivants :
   a. Créer une liste des 4 cases adjacentes en augmentant le compteur
   b. Si la case est un mur ou si elle existe déjà dans la liste principale, la
      retirer
   c. Ajouter toutes les cases restantes à la fin de la liste principale
4. Réaliser a, b et c jusqu'à tomber sur l'élément de début
5. Démarrer de cet élément et prendre l'élément adjacent avec le compteur le

plus bas

6. On définit ainsi le chemin le plus cour

Pas connaitre complexité graphe

▼ Algo de recherche adversariale

Minimax

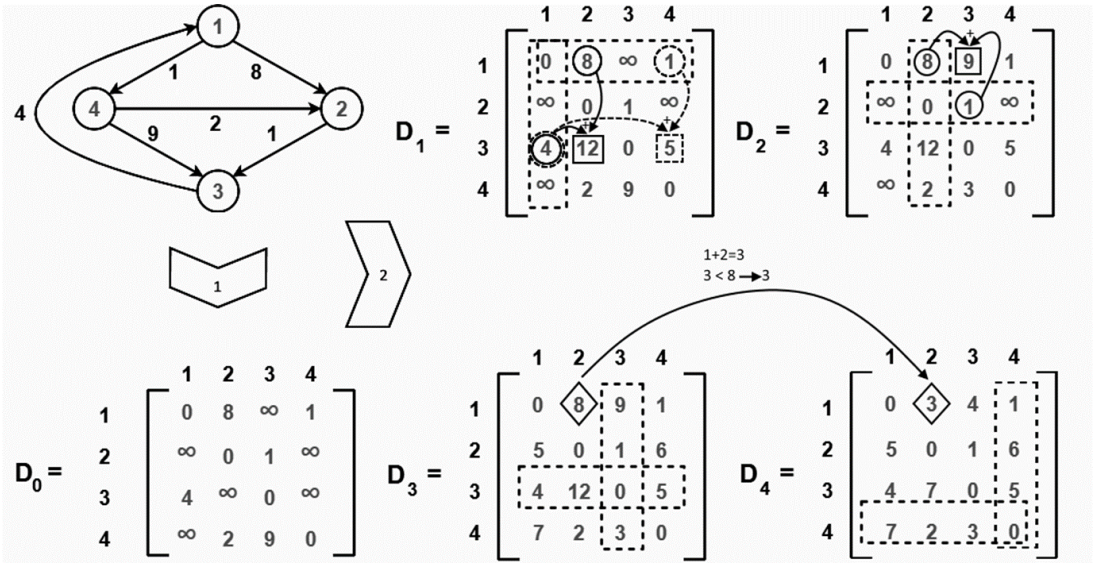
- Règle pour les jeux a somme nulle
- Objectif : minimiser la perte potentielle maximale
- Jeux a deux joueurs alternatifs
- Le joueur est appelé joueur maximisant
- L’opposant est appelé joueur minimisant

▼ Algo de jeux puzzle

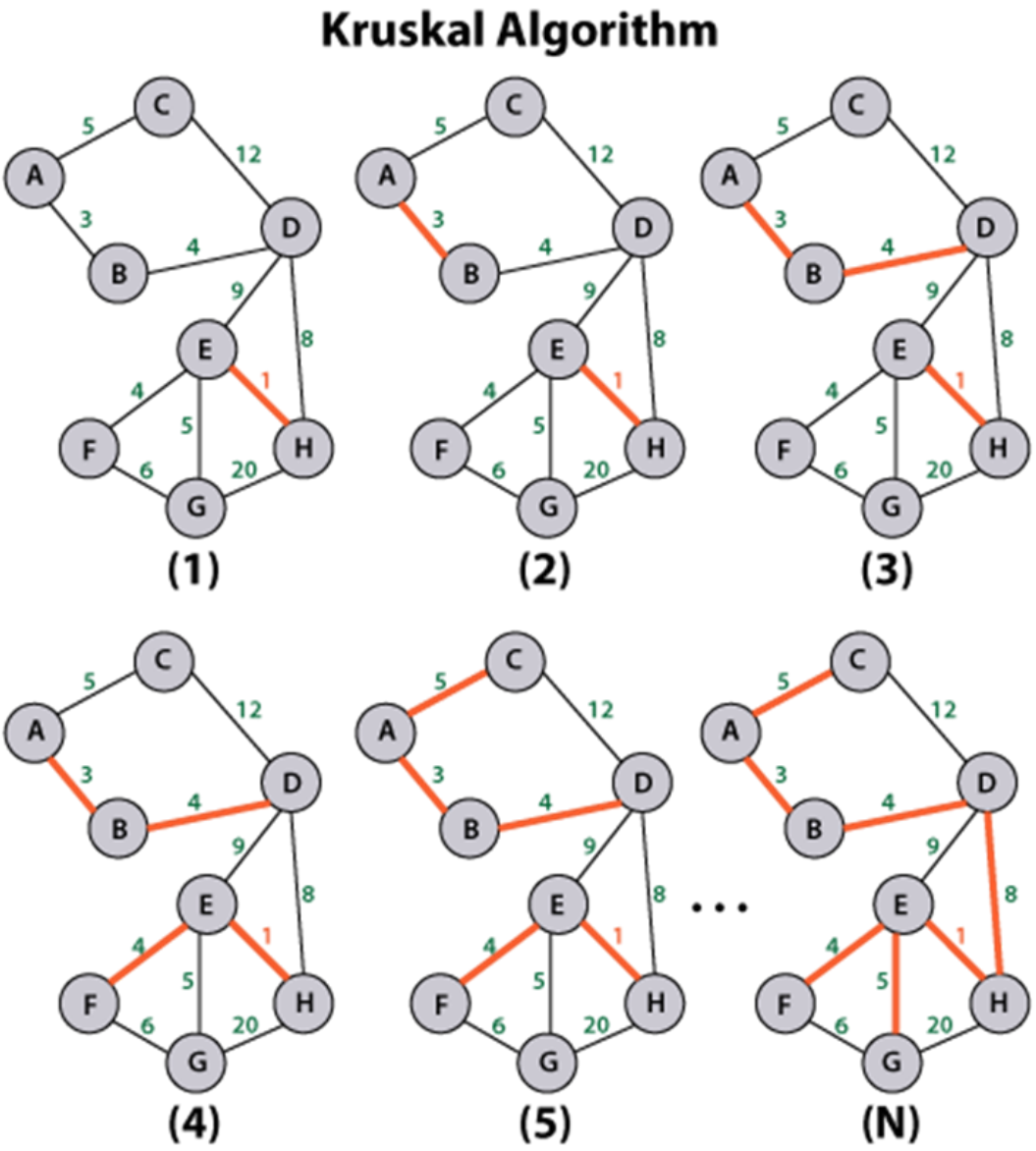
jeu ou le défi est de trouver une solution optimale à un problème donnée

REVOIR POWERPOINT SCÉANCE 3-4 POUR SCHÉMA

▼ FLOYD-WARSHALL



▼ KRUSKAL



▼ MINIMAX

VOIR PWP SCÉANCE 5

Backtraking

- Construire une solution de manière récursive, étapes par étape en retirant les solutions

Chiffrement

- Objectif principaux :
  - Confidentialité
    - peut pas la lire sauf ceux concerné

- Intégrité
  - information envoyé = information récupérer
- Authenticité
  - possible de vérifier que l'information vient du bon endroit

Technique de bases

- Substitution
  - Cesar
- Transposition (intervertir des valeur)
- Par bloc
  - DES, AES
- Par flux

▼ Chiffrement symétrique

- DEs (Data Encryption Standard)
  - Fonctionne par blocs dans lequel on implique une clé de 56 bits et qui ressort un cipher texte de 64 bits puis divise en 2 les blocs et les passez en 16 rounds de passage de clé et ensuite on a une étape finale qui ressort le texte 64 bits
- 3DES (triple DES)
  - 3 Liens entre le reverse et le normal
- AES (Advanced Encryption Standars)
- Blowfish
- Twofish

Mode de chiffrement

- ECB (electronic codebook)
  - Divise les données en blocs de taille fixe.

    - Vulnérable aux attaques de type dictionnaire car des blocs identiques sont chiffrés de la même manière
- CBC (Cipher Block Chaining)
    - Bloc de données combiné avec le précédent.
    - Cela rend l’attaque de type dictionnaire plus difficile
- CTR (counter)
- OFB (Output Feedback**)**

Problèmes courants

- Échange de clés
- Distribution des clés
- Rotations des clés

Solution

- Chiffrement de clé
- PKI (public key infrastructure : la PKI est un système de chiffrement asymétrique qui permet d’authentifier et de sécuriser les communications électroniques. La PKI utilise des certificats numérique pour vérifier l’identité des parties et assurer l’intégrité des données
- Gestions de clés centralisée : les organisations peuvent utiliser des systèmes de gestion de clés centralisés pour gérer les clés de chiffrement. Ces systèmes peuvent faciliter l'échange, la distribution et la rotation des clés.

Vulnérabilités

- Brute Force
- Attaque par dictionnaire
    - Utilisation d’une liste de mots courants. Cette attaque peut être efficace pour des clés courtes ou faibles
- Attaque par analyse différentielle

    - Analyse des différences entre les textes clairs et les textes chiffrés pour trouver des patterns ou des vulnérabilités dans le chiffrement
- Attaque par force brute en temps réel
    - Attaque interceptant le message chiffré et essayant de déchiffre le message en temps réel en utilisant différentes clés jusqu'à ce que le message chiffré soit déchiffré avec succès

Améliorer la sécurité

- Augmentation de la longueur de la clé
- Utilisation des clés aléatoires
- Utilisation de modes de chiffrement plus sur
- Utilisations de clés uniques

Utilisation

- Chiffrement de fichiers
- Chiffrement de disques
- Chiffrement de base de données
- Chiffrement de communications

▼ Chiffrement asymétrique

Chiffrement asymétrique va utiliser un couple de clés :

- Une clé publique (p)
- Clé privé (s)

La clé publique → chiffrer le message

la clé privée → le déchiffrer

il y a donc une fonction de chiffrement f et une fonction de déchiffrement g :

g(f(message,p)s) = message

Algorithmes :

- RSA
- Chiffrement El Gamal
- Chiffrement de **PAS FINI**

RSA

- Rivest Shamir Adleman (RSA) est basé sur la difficulté de factoriser de grands nombres entiers en facteur premier
- On peut définir le RSA avec les informations suivantes :
  - Clé publique (n,e)
  - Clé privée (d)
  - Fonction de chiffrement
  - Fonction de déchiffrement
- Pour générer une clé RSA :
  - On choisit 2 nombres premiers distincts p et q
  - On calcule le module de chiffrement n = p*q
  - On calcule l'indicatrice d'Euler : $\varphi(n)=(p-1)(q-1)$
  - On choisit un nombre premier $e<\varphi(n)$ premier avec $\varphi(n)$
  - On calcule le nombre entier $d= e$^(−1) $mod\ \varphi(n)$
- Attaques
  - Brute force sur p & q
  - Attaque de Wiener
  - attaque de Hastad
  - time attack

- Adaptive chose cyphertext attack

▼ **RegEx**

Une expression régulière est une chaîne de caractères qui décrit un ensemble de chaînes de caractères

Les ensemble :

- Représenté par des []
  - [a-z] = minuscule
  - [0-9] = chiffre
  - [”&é’ (et le reste)] = caractère spéciaux (a compléter au besoin)
  - [^a] ce qui n’est pas dans mon ensemble
    - [a-zA-z0-9!?] tout en gros

Raccourcis d’ensemble

- \w : lettre + chiffre + _
- \d : chiffres
- \s : caractères blancs
- . : n’importe quel caractère
- Chaque raccourci en maj donne la négation

Opérateurs

- a|b : les a OU les b
- Opérateurs de quantité
  - ? : 0 ou 1 élément
  - * : 0 ou plusieurs éléments
  - + : au moins 1 éléments
  - {n,m} : un nombre défini d’éléments
- Exemple
  - \b\w{4,6}\b

```
Tel : \+32 \d{3}\/\d{2}.?\d{2}.?\d{2}
```

/