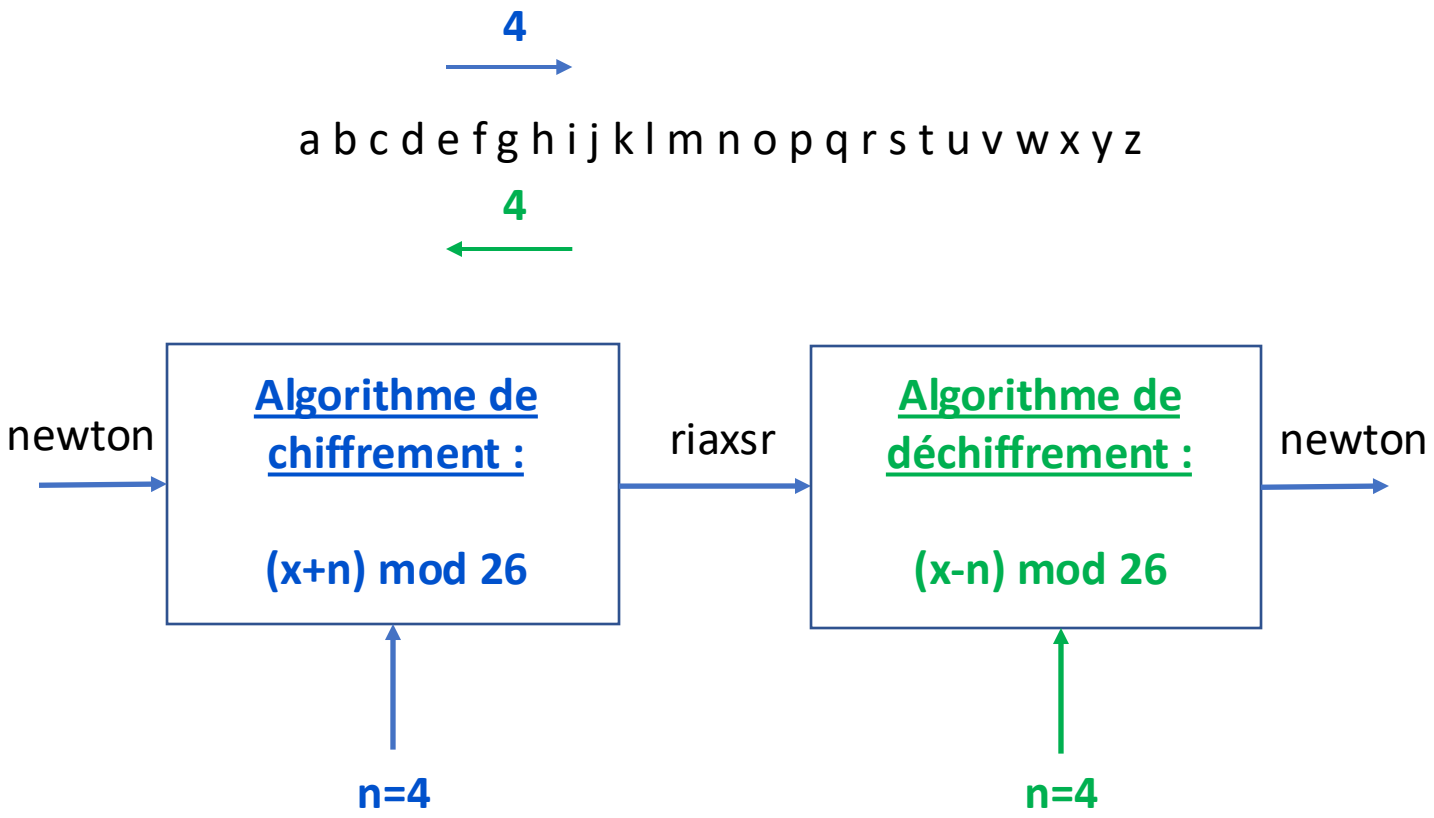
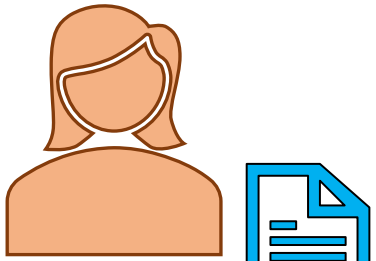
PRINCIPE DU CHIFFREMENT/DÉCHIFFREMENT SYMETRIQUE

On appelle ce chiffrement : code de césar ou chiffrement par décalage.



N est appelé la **clé de chiffrement**.
Dans le cas du chiffrement **symétrique**, la clé est identique pour le chiffrement et le déchiffrement.

CHIFFREMENT/DÉCHIFFREMENT SYMETRIQUE



ALICE

BOB

2. TRANSFERT DU FICHIER CHIFFRÉ

3. DÉCHIFFREMENT DU FICHIER AVEC LA CLÉ

1.CHIFFREMENT DU FICHIER AVEC LA CLÉ

CLÉ SYMÉTRIQUE UNIQUE

+

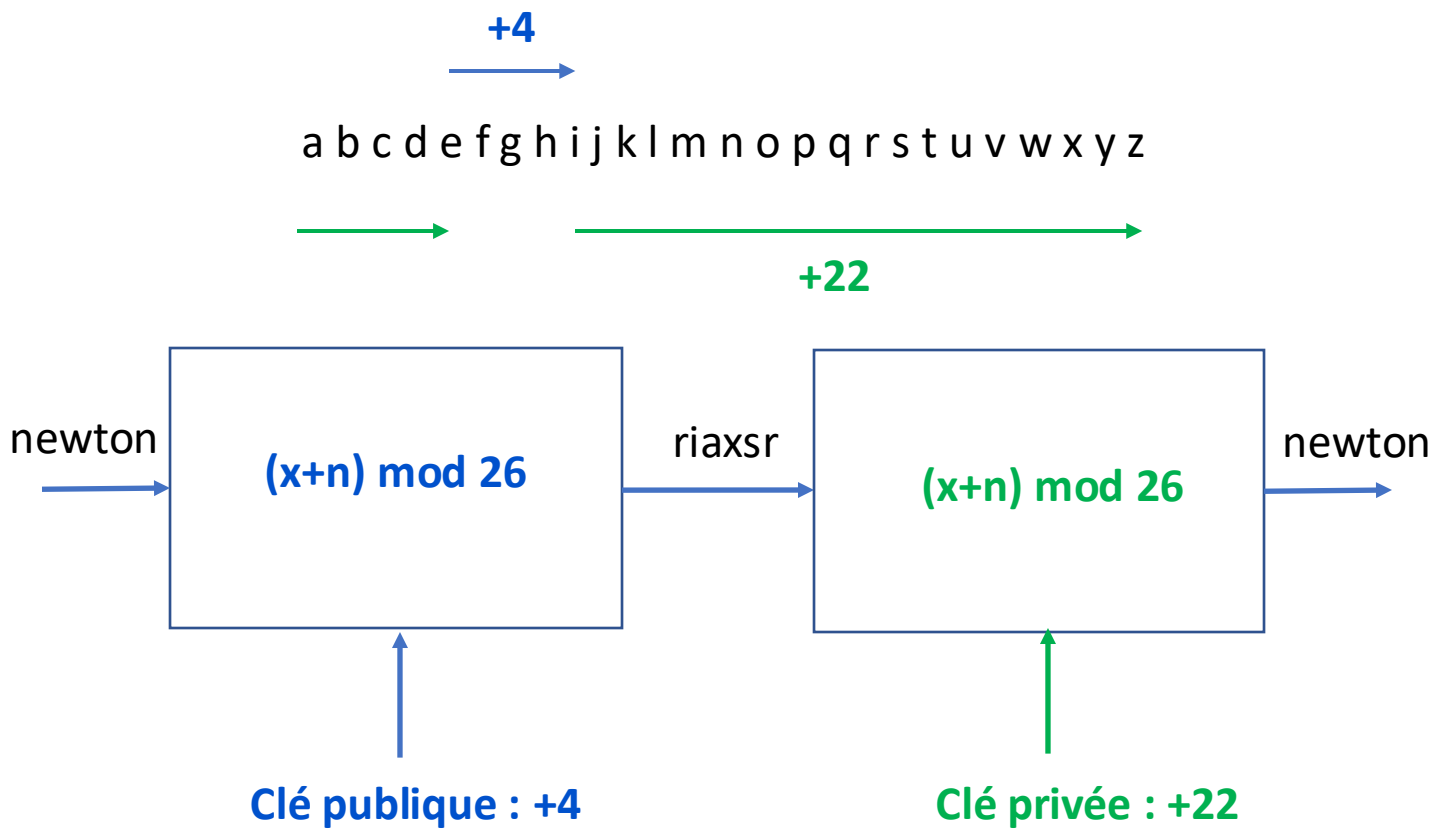
+

CLÉ SYMÉTRIQUE UNIQUE

Cette technique est très simple, rapide et peu gourmande en ressource CPU.

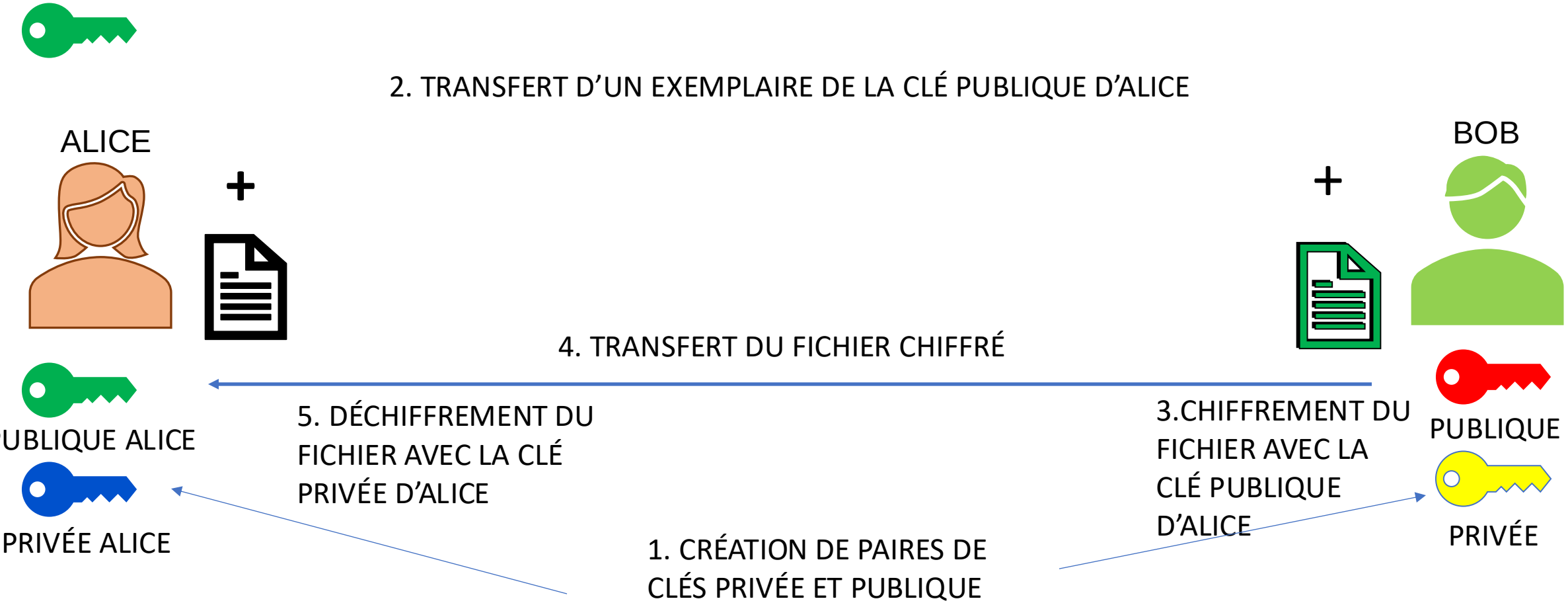
RISQUE DE VOL DE LA CLÉ DE CHIFFREMENT LORS DU PARTAGE DE LA CLÉ ENTRE LES 2 UTILISATEURS.

PRINCIPE DU CHIFFREMENT/DÉCHIFFREMENT ASYMETRIQUE



Les deux clés sont créées ensemble, on parle de paires de clés asymétriques

CHIFFREMENT/DÉCHIFFREMENT ASYMETRIQUE



Avantage : la technique de transfert est sécurisée. Seule la clé publique est visible. La clé privée n'est pas transmise.
Inconvénient : lent et gourmand en ressource CPU.

CODE BASH DE CHIFFREMENT/DÉCHIFFREMENT DE CÉSAR

```
#!/bin/bash

# Fonction pour le chiffrement/déchiffrement de César
cesar() {
   local texte="$1"      # Texte à traiter
   local decalage="$2"    # Décalage
   local mode="$3"       # Mode : "chiffre" ou "dechiffre"

   # Ajuste le décalage en fonction du mode
   if [ "$mode" == "dechiffre" ]; then
      decalage=$((26 - decalage)) # Inverse le décalage pour déchiffrer
   fi

   # Initialise la chaîne de sortie
   local resultat=""

   # Boucle sur chaque caractère du texte
   for ((i=0; i<${#texte}; i++)); do
      char="${texte:$i:1}"

      # Vérifie si le caractère est une lettre majuscule ou minuscule
      if [[ "$char" =~ [A-Z] ]]; then
         base=65 # Code ASCII de 'A'
         code=$(printf "%d" "'$char")
         new_code=$(( (code - base + decalage) % 26 + base ))
         resultat+=$(printf "\\$(printf "%o" "$new_code")")
      elif [[ "$char" =~ [a-z] ]]; then
         base=97 # Code ASCII de 'a'
         code=$(printf "%d" "'$char")
         new_code=$(( (code - base + decalage) % 26 + base ))
         resultat+=$(printf "\\$(printf "%o" "$new_code")")
      else
         # Garde les caractères non alphabétiques inchangés
         resultat+="$char"
      fi
   done

   echo "$resultat"
}
```

```
# Exemple d'utilisation
texte="HELLO"
decalage=3

# Chiffrement
chiffre=$(cesar "$texte" "$decalage" "chiffre")
echo "Texte chiffré : $chiffre"

# Déchiffrement
dechiffre=$(cesar "$chiffre" "$decalage" "dechiffre")
echo "Texte déchiffré : $dechiffre"
```