Crypto101 Express

Block ciphers

DaVinciCode

15/06/20

Rappel

- OTP pas pratiques
  - taille des données = taille de la clef
  - problème de transmission des clés

Rappel

- OTP pas pratiques
  - **taille des données = taille de la clef**
  - problème de transmission des clés

Chiffrement par bloc

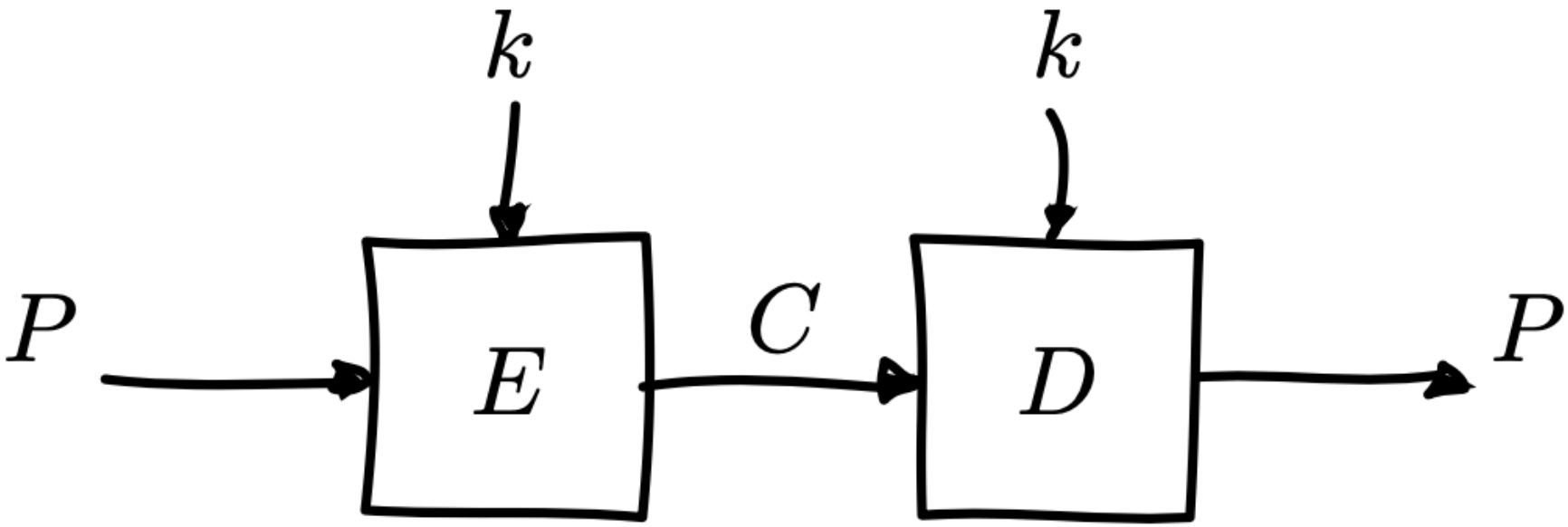
Définition

Algorithmes qui permettent de chiffrer/déchiffrer des blocs de taille fixe (*e.g.* 16 bytes)

$$C = E(k, P)$$

$$P = D(k, C)$$

Graphique



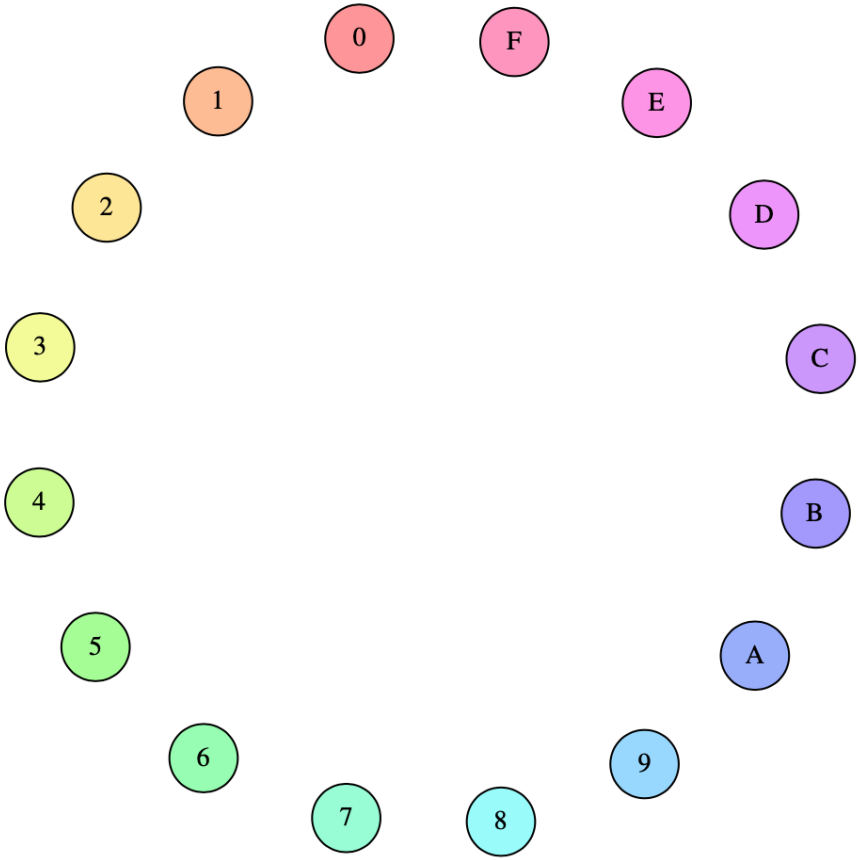
Remarque

Les algorithmes de chiffrement par bloc font partie de la cryptographie symmétrique du fait que la même clef est utilisée pour le chiffrement et le déchiffrement

Comment ça fonctionne?

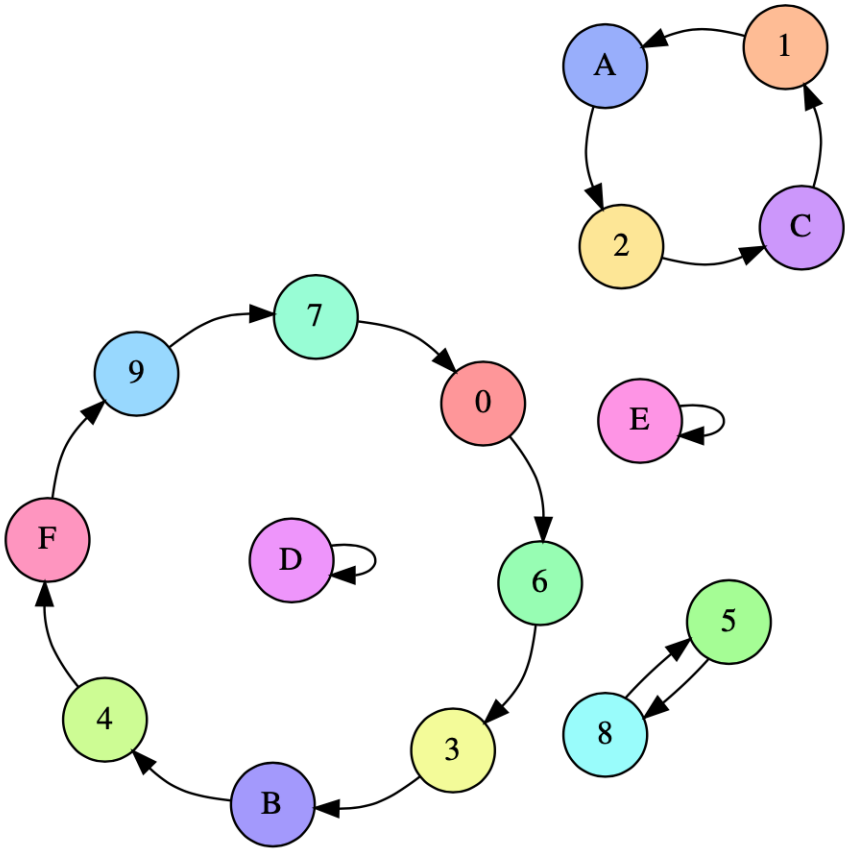
- imaginons que l'on veuille chiffrer un bloc de 4 bits
  - peut être représenté en hexa
    - $0000_2 = 0_{16}$
    - $0001_2 = 1_{16}$
    - ...
    - $1010_2 = a_{16}$
    - $1011_2 = b_{16}$
    - ...
    - $1111_2 = f_{16}$
- une clef $\Rightarrow$ un ensemble de permutations

Comment ça fonctionne?



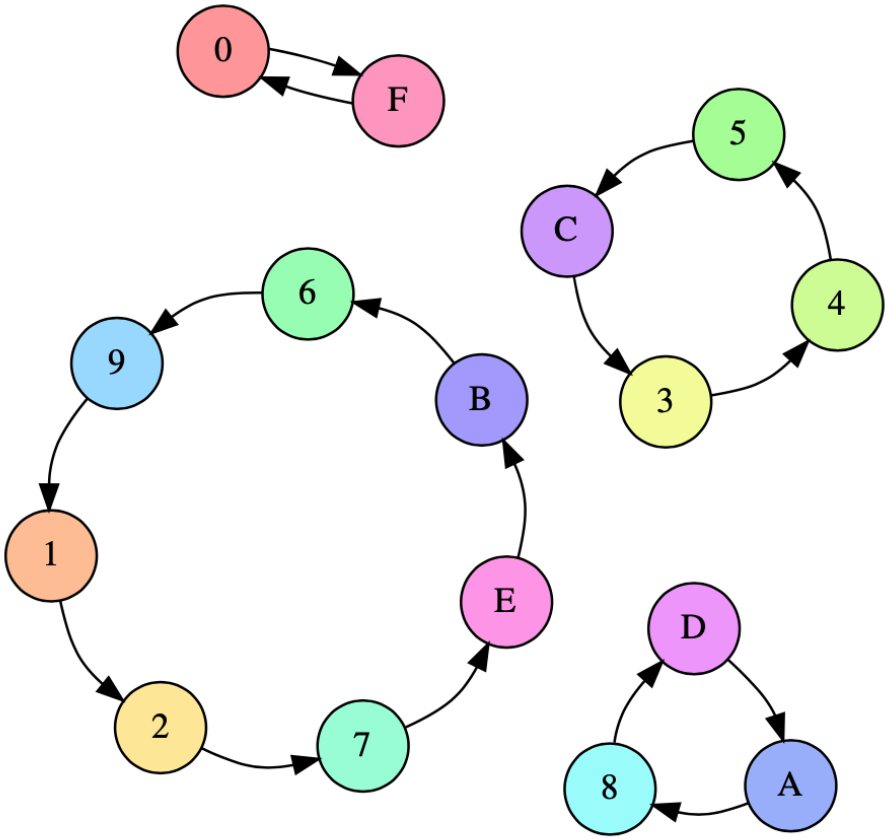
Comment ça fonctionne?

- Chiffrement avec k_1



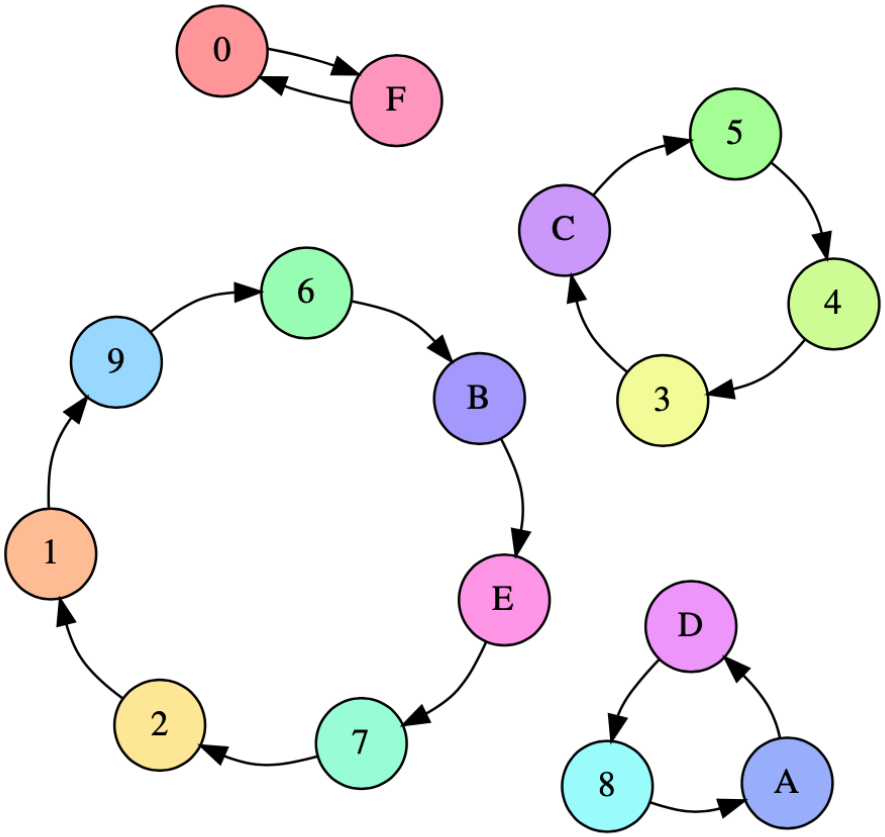
Comment ça fonctionne?

- Chiffrement avec k_2



Comment ça fonctionne?

- Déchiffrement avec k_2



DES

- Data Encryption Standard, standardisé en 1977
- taille d'un bloc: 64 bits
- taille d'une clef: 64 bits
- taille effective d'une clef: 56 bits (8 bits de parité)
- bruteforcable en un jour (il semblerait)

```
2**56
```

```
## 72057594037927936
```

DES sur Python

```
from Crypto.Cipher import DES

key = b'13371337'
cipher = DES.new(key, DES.MODE_ECB)
ciphertext = cipher.encrypt(b'deadbeef')
plaintext = cipher.decrypt(ciphertext)
print(ciphertext)
```

```
## b'\xd5r\x12\t\x86shQ'
```

```
print(plaintext)
```

```
## b'deadbeef'
```

3DES

$$C = E_{DES}(k_1, D_{DES}(k_2, E_{DES}(k_3, P)))$$
$$P = D_{DES}(k_3, E_{DES}(k_2, D_{DES}(k_1, C)))$$

- tentative d'étendre la vie de l'algorithme DES
- si $k_1 \neq k_2 \neq k_3 \Rightarrow 168$ bits
- si $k_1 = k_3 \Rightarrow 112$ bits
- si $k_1 = k_2 = k_3 \Rightarrow$ DES
- ça reste un mauvais choix

```
2**168
```

```
## 374144419156711147060143317175368453031918731001856
```

```
2**112
```

```
## 5192296858534827628530496329220096
```

3DES sur Python

```
from Crypto.Cipher import DES3

key = b'13371337deadbeefbaddcafe'
# key = key1||key2||key3 ou key1||key2
cipher = DES3.new(key, DES3.MODE_ECB)
ciphertext = cipher.encrypt(b'universe')
plaintext = cipher.decrypt(ciphertext)
print(ciphertext)

## b'\x11\t\x8aC\nv\xce|'
print(plaintext)

## b'universe'
```

Pourquoi pas 2DES?

- vulnérable aux attaques meet-in-the-middle

$$C = ENC_{k_2}(ENC_{k_1}(P))$$

$$P = DEC_{k_1}(DEC_{k_2}(C))$$

$$\begin{aligned} & C = ENC_{k_2}(ENC_{k_1}(P)) \\ \iff & DEC_{k_2}(C) = DEC_{k_2}(ENC_{k_2}[ENC_{k_1}(P)]) \\ \iff & DEC_{k_2}(C) = ENC_{k_1}(P) \end{aligned}$$

- la sécurité du 2DES n'est que faiblement supérieure à celle du DES (2^{57} *vs* 2^{56})

AES

- Advanced Encryption Standard (ou Rijndael)
- standardisé en 2002
- taille d'un bloc: 128 bits
- taille d'une clé: 128, 192, ou 256 bits

Pour aller plus loin

- https://cryptohack.org/challenges/aes/
- Implémentation simplissime de l'algorithme AES en Python

Toujours des inconvenients

- comment faire pour chiffrer un message de longueur indeterminée ?
- problème de transmission des clés

Des questions?