

# Crypto101 Express

## Stream ciphers

DaVinciCode

25/06/20



- Chiffrement par bloc
  - comment faire pour chiffrer un message de longueur indéterminée?
  - problème de transmission des clés

- Chiffrement par bloc
  - **comment faire pour chiffrer un message de longueur indéterminée?**
  - problème de transmission des clés

# Avec le chiffrement par blocs

- diviser le message par blocs et les chiffrer indépendamment

abcdefgh	ijklmno	pqrstuvw	...
↓	↓	↓	
APOHGMMW	PVMEHQOM	MEEZSNFM	...

- on appelle ce mode d'opération le mode ECB (Electronic codebook)
- 

$$C_i = E_k(P_i)$$

- comment faire si on veut chiffrer un message si sa taille n'est pas un multiple de la taille du bloc?
  - on rajoute un padding (rembourrage)

# Padding

- ajouter des zeros (le message ne peut donc pas finir par un byte nul)
- PKCS#7: si il nous manque 5 bytes pour que la taille soit un multiple de la taille du bloc, on rajoute: 05 05 05 05 05
  - si on veut vraiment finir le message avec 5 bytes 05, on rajoute un bloc entier de padding (08 08 08 08 08 08 08 08)

# Désavantages du mode ECB

- deux blocs identiques seront chiffrés de la même manière

abcdefgh    abcdefgh    abcdefgh    ...  
↓            ↓            ↓  
APOHGMMW    APOHGMMW    APOHGMMW    ...

# Démo - Message

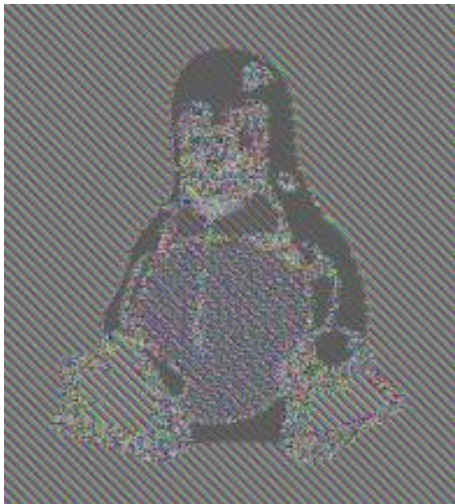




# Démo - Chiffrement idéal

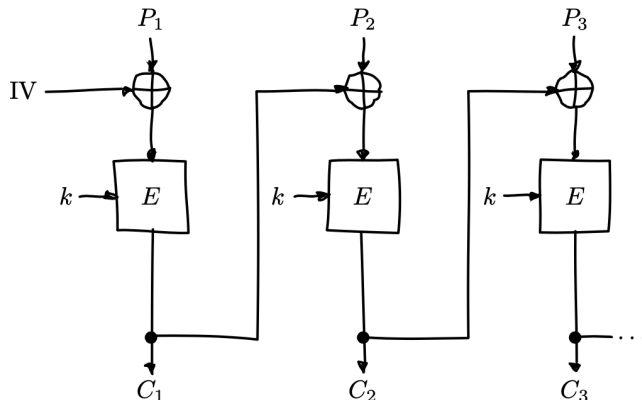


# Démo - Chiffrement avec ECB



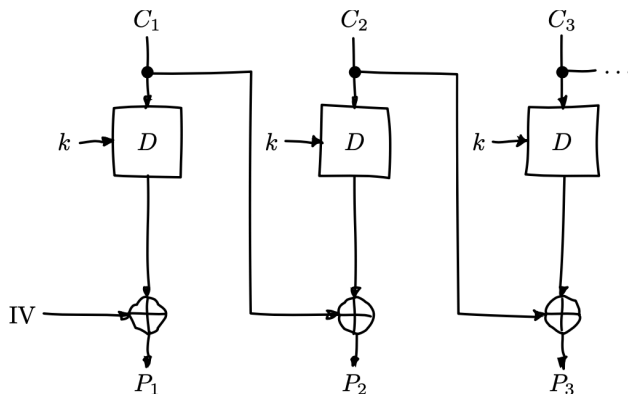
- Cipher block chaining
- $C_i = E_k(P_i \oplus C_{i-1})$
- On XOR chaque bloc  $P_i$  par le bloc chiffré  $C_{i-1}$  précédent pour brouiller les motifs
  - deux blocs égaux ne seront pas chiffrés de la même manière

# Graphique - Chiffrement avec CBC



- $C_0 = IV$  (initialization vector)

# Graphique - Déchiffrement avec CBC



- $P_i = D_k(C_i) \oplus C_{i-1}$

# Initialization vector

- il est envoyé avec le message chiffré
- il doit être imprévisible, **mais pas secret**
  - il ne doit surtout pas être égal à la clef

# Attaques si l'IV est prévisible

- imaginons le site d'une banque qui utilise le mode CBC pour chiffrer les données de ses clients
  - pour simplifier, 1 solde  $\Rightarrow$  1 bloc
  - on peut actualiser notre solde
- base de données:

Client	Solde
Alice	$C_A = E(k, IV_A \oplus P_A)$
Mallory	$C_M = E(k, IV_M \oplus P_M)$
Bob	$C_B = E(k, IV_B \oplus P_B)$

# Attaques si l'IV est prévisible

- Mallory est maline
  - elle arrive à prédire les IV qui ont été utilisés pour chiffrer les données pour chaque client ( $IV_A, IV_M, IV_B$ )
  - elle a accès à la base de données chiffrées
- ① elle actualise son solde  $P_M = IV_M \oplus IV_A \oplus G$
- ② la banque actualise la base de données:  $C_M = E(k, IV_M \oplus P_M)$   
 $\Leftrightarrow C_M = E(k, IV_M \oplus (IV_M \oplus IV_A \oplus G))$   
 $\Leftrightarrow C_M = E(k, IV_A \oplus G)$
- ③ si  $C_M = C_A$ , alors Mallory a trouvé le solde d'Alice ( $G$ )

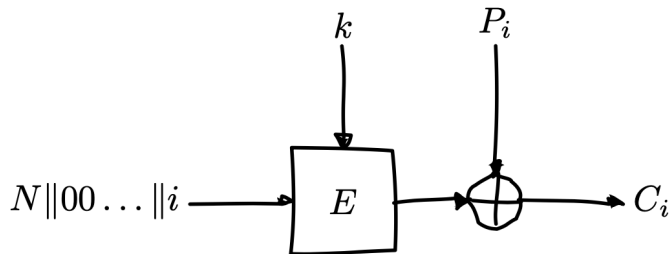
## Remarque

Cette banque aurait aussi du avoir une clé  $k$  différente pour chaque client



- **counter** mode
- nonce (number used once) à ne pas réutiliser
- $C_i = P_i \oplus E(k, N || 00... || i)$

# CTR mode - Graphique



- **native** stream cipher
- état de l'art
- créé par Dan Bernstein
- ARX (add, rotate, XOR) design
  - $x \leftarrow x \oplus (y \boxplus z) \lll n$
  - modular addition:  $\boxplus$
  - rotation:  $\lll$

# Salsa20 sur Python (chiffrement)

```
from Crypto.Cipher import Salsa20

plaintext = b'TOP_SECRET'
secret = b'\x13\x37'*16 # 32-byte key
cipher = Salsa20.new(key=secret)
msg = cipher.nonce + cipher.encrypt(plaintext)
print(msg[:8])
```

```
## b' /\xdcJ\xdB7\xce\xa3'
```

```
print(msg[8:])
```

```
## b'\x0bz>\xd8M\xba\x95\xce\xb6\x10'
```


# Salsa20 sur Python (déchiffrement)

```
from Crypto.Cipher import Salsa20

secret = b'\x13\x37'*16
msg_nonce = b't\xe6\xf\x142c\xe40'
ciphertext = b'\xe8\x8d\xfb\x9c\xce[@\xb5\x1aP'
cipher = Salsa20.new(key=secret, nonce=msg_nonce)
plaintext = cipher.decrypt(ciphertext)
print(plaintext)

## b'TOP_SECRET'
```

# Toujours des problèmes >:(

- chiffrer n'importe quel message de manière sécurisée 
- problème de transmission des clefs 