

Crypto101 Express

Exclusive OR

DaVinciCode

11/6/2020





- Louis
- A5 IOS
- Vice-président de DaVinciCode

- livre Crypto101 (crypto101.io)
- livre *Introduction to Mathematical Cryptography*
- challenges Cryptohack (cryptohack.org)

Binaire

- Tout nombre peut être écrit comme la somme de puissances de 2
- Par exemple, 27 s'écrit

1	1	0	1	1
2^4	2^3	2^2	2^1	2^0
16	8	0	2	1

```
0b11011
```

```
## 27
```

```
bin(27)
```

```
## '0b11011'
```

Octet et hexa

- base 16 (0, ..., 9, a, ..., f)
- 16 chiffres
- si on veut représenter 1 octet en hexa: $2^8 = 256$ il nous faut deux chiffres ($16 \times 16 = 256$)
- (*un chiffre en base 16 représente 4 bits*)

```
bytes.fromhex('deadbeef') == b'\xde\xad\xbe\xef'
```

```
## True
```

```
bytes.fromhex('6465616462656566') == b'deadbeef'
```

```
## True
```

ASCII

- acronyme pour American Standard Code for Information Interchange
- codage de caractères: chaque caractère peut être représenté par un nombre

```
[chr(i) for i in range(65, 75)]
```

```
## ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
```

Objectif de la cryptographie

- tout peut être représenté comme un nombre
- Alice veut envoyer un nombre à Bob de manière secrète
- on cherche à créer un modèle mathématique

OU exclusif (XOR)

- opérateur noté \oplus

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

- pour les mots binaires, on fait l'opération bit par bit: $1100_2 \oplus 0101_2 = 1001_2$

```
bin(0b1100 ^ 0b0101)
```

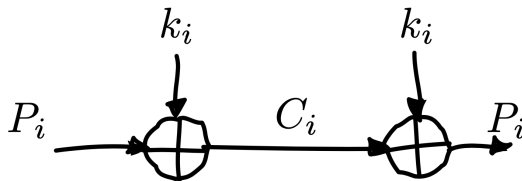
```
## '0b1001'
```


Quelques propriétés du XOR

- $A \oplus A = 0$
- $A \oplus 0 = A$
- $A \oplus 1 = \overline{A}$ ($\overline{1001_2} = 0110_2$)
- $A \oplus B = B \oplus A$ (commutativité)
- $A \oplus (B \oplus C) = (A \oplus B) \oplus C$ (associativité)
- $B \oplus A \oplus B = A \oplus (B \oplus B) = A$

One Time Pad

- en français masque jetable (mdr)
- très simple: message \oplus clé
- offre une sécurité théorique absolue (prouvé par Claude Shannon en 1949)



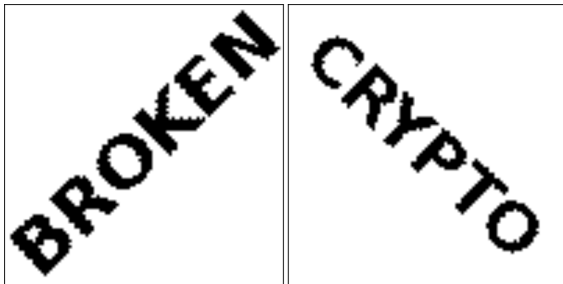
Limitations

- la clef doit être vraiment aléatoire (sinon, k_i peut être prédit)
- la clef ne doit pas être réutilisée

$$c_1 \oplus c_2 = (p_1 \oplus k) \oplus (p_2 \oplus k)$$

$$c_1 \oplus c_2 = p_1 \oplus p_2$$

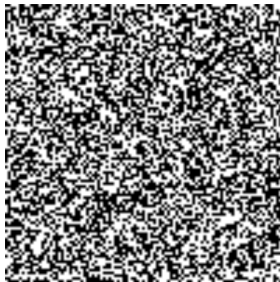
- on souhaite envoyer deux images



(a) First plaintext.

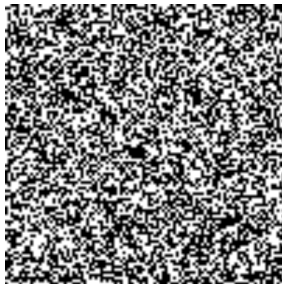
(b) Second plaintext.

- avec la même clef

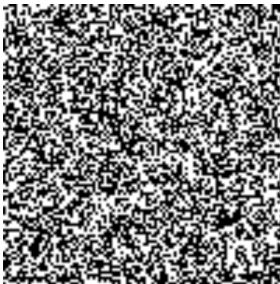


(e) Reused key.

- on envoie alors $P_i \oplus k = C_i$



(c) First ciphertext.



(d) Second ciphertext.



(f) XOR of ciphertexts.

The end?

The end?


- Peu pratique
 - 50 GB¹ de données \Rightarrow 50 GB de clef
 - Problème de transmission des clés

¹GB: gigabytes ou gigaoctets (10^9 octets) à ne pas confondre avec Gb (gigabits)



Application

- <https://cryptohack.org/challenges/misc/>
- Challenge “Gotta go fast”
- Objectif: trouver une chaîne de caractères de la forme `crypto{...}` (*flag*)

Analyse du code source

- la clef n'est pas réutilisée 

Analyse du code source

- la clef n'est pas réutilisée 
- la clef n'est pas aléatoire 
 - $k = \text{sha256}(t)$ (avec t le nombre de secondes passées depuis le 1er janvier 1970)

- deux options:
 - {"option": "get_flag"} retourne le flag masqué
 - {"option": "encrypt_data"} retourne le message donné en paramètres masqué