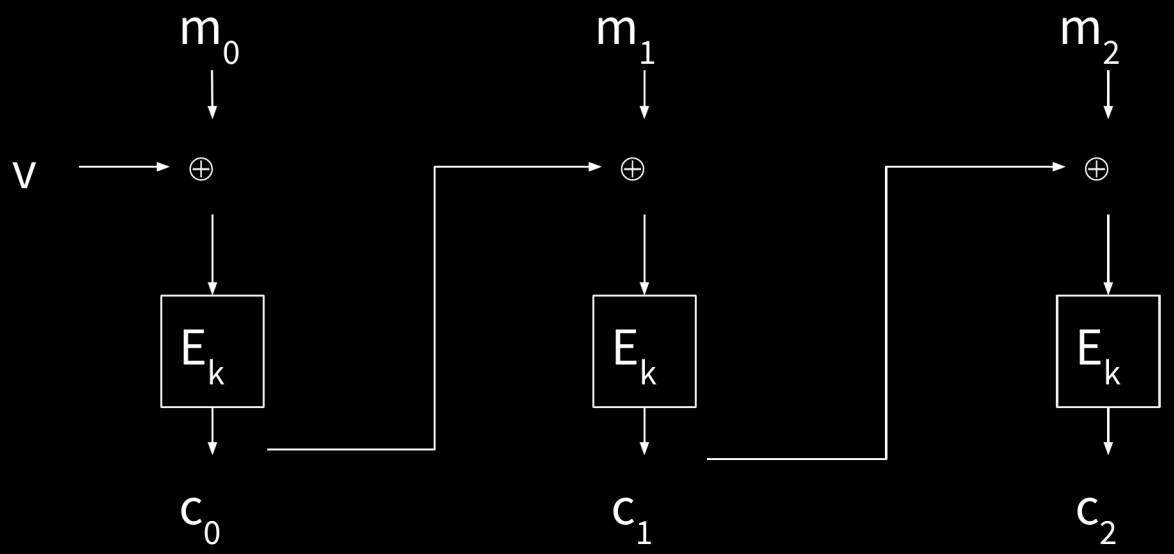
Exercice ‘plus dur’

c. Montrer que le mode opératoire CBC n’assure pas la sécurité sémantique pour des messages suffisamment longs.



Exercice ‘plus dur’

- On considère deux blocs égaux: forte probabilité d’avoir deux blocs égaux pour des messages de plus de $2^{n/2}$ blocs de n bits (cf TD1).

  $C_i = C_j$ avec $i \neq j$

  Alors: $C_i = E_k (C_{i-1} \oplus m_i)$ et $C_j = E_k (C_{j-1} \oplus m_i)$

Or E_k est une fonction de chiffrement, c’est une permutation, on a:

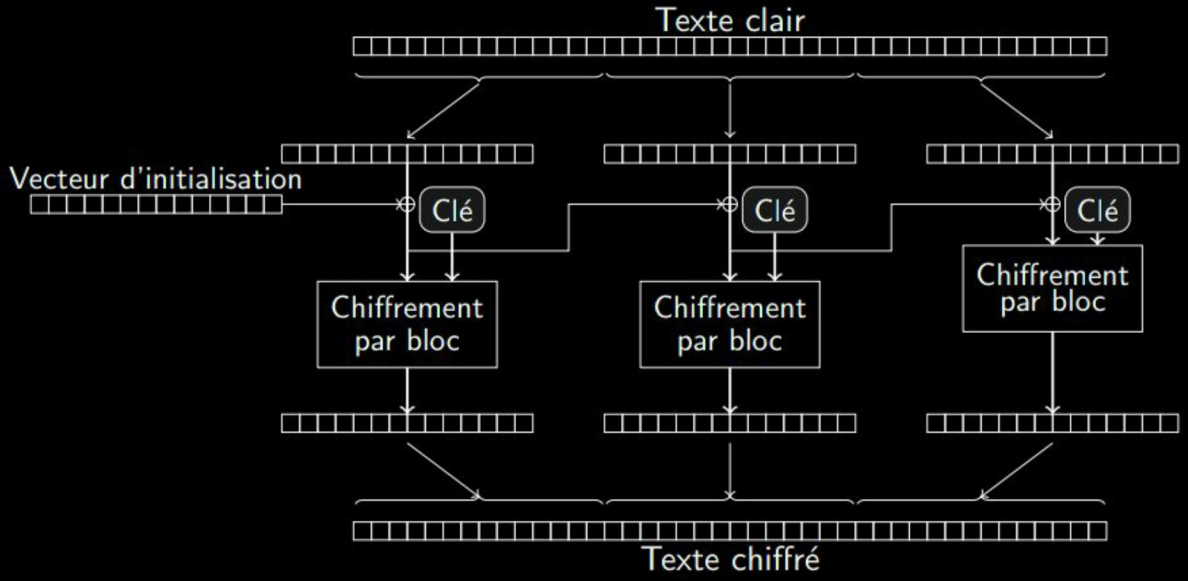
$E_k(a) = E_k(b) \Leftrightarrow a = b$

Donc $C_{i-1} \oplus m_i = C_{j-1} \oplus m_i$

- Si on connaît deux clairs et qu'on sait que C est le chiffré de l'un d'entre eux, il suffit de regarder lequel des deux clairs satisfait l'égalité précédente.
- On a facilement deux blocs égaux (répétition) si on chiffre un message de plus de $2^{n/2}$. Le chiffrement par bloc CBC n'assure pas la sécurité sémantique à condition que les messages à chiffrer soient très longs.

Exercice 2: Mode opératoire CBC*

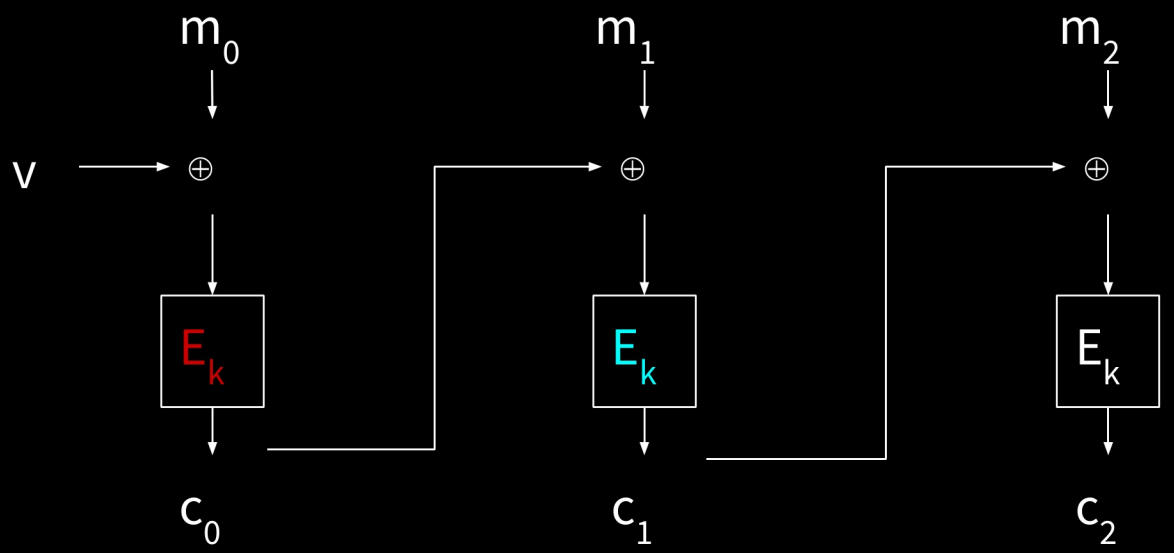
- Inconvénient du mode CBC: intrinsèquement séquentiel => ne permet pas de paralléliser les opérations de chiffrement.
- On considère donc le chiffrement modifié CBC* qui permet d'effectuer plusieurs opérations de chiffrement & déchiffrement en //:



Exercice 2: Mode opératoire CBC*

1. Décrire comment le déchiffrement est effectué pour le mode CBC*.

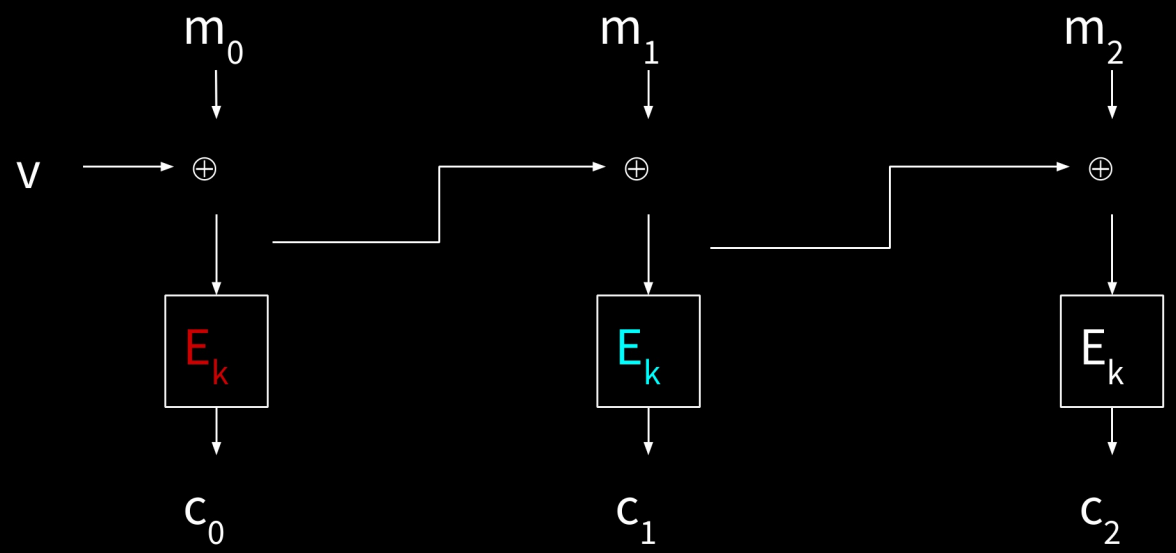
- On rappelle le chiffrement CBC et on s'intéresse à la parallélisation:



C’est bien non parallélisable, on ne peut pas passer dans E_k sans être passé dans E_k

Exercice 2: Mode opératoire CBC*

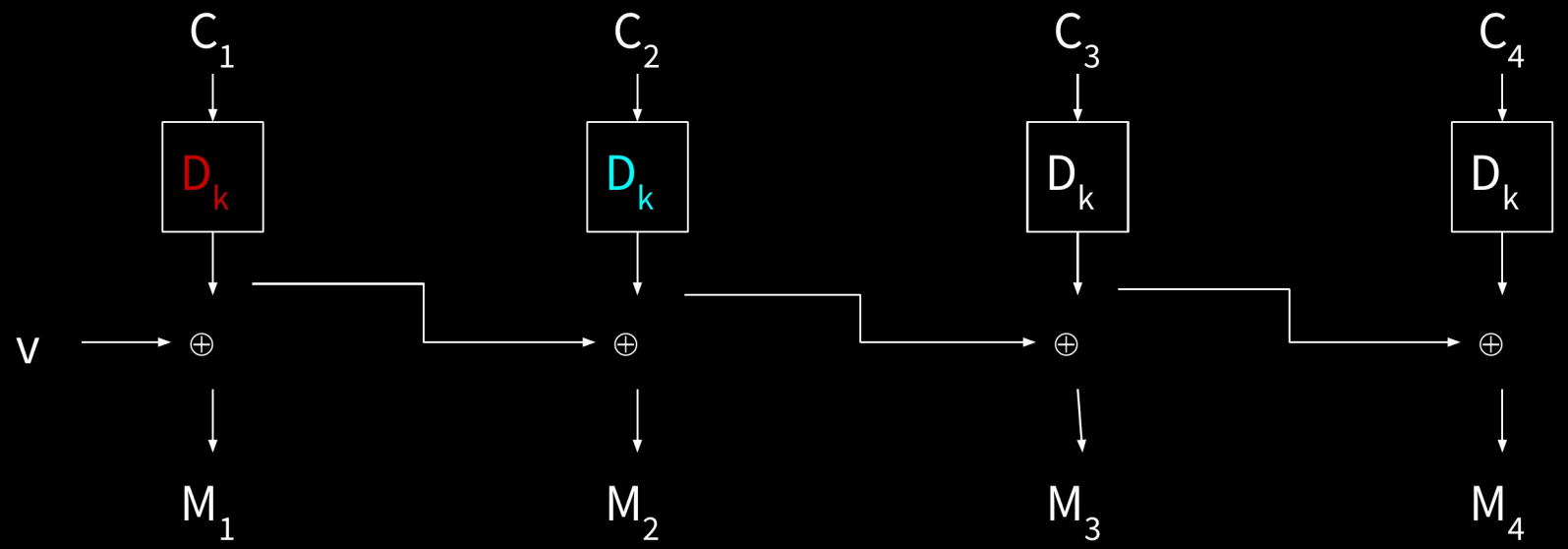
- On modifie CBC pour obtenir un chiffrement parallélisable, CBC*:



➔ Avec un peu de précalcul, on peut chiffrer tous les blocs en même temps!

Exercice 2: Mode opératoire CBC*

- Vérifions que le déchiffrement est aussi parallélisable:

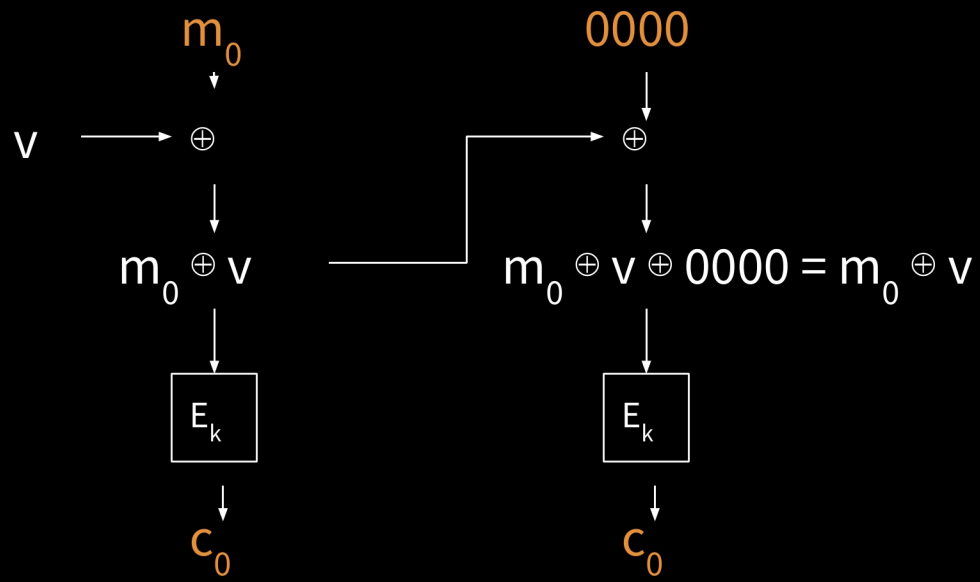


Là encore on peut passer dans tous les blocs de déchiffrement en même temps. Le déchiffrement CBC* est bien parallélisable.

Exercice 2: Mode opératoire CBC*

2. Montrer que ce mode opératoire n'assure pas la sécurité sémantique.

On choisit $M_0 = m_0\ 0000$ et $M_1 = m_0 m_1$ avec $m_1 \neq 0000$ (à tester par vous même)



Si le message clair est de la forme $m_0\ 0000$ alors le chiffré C se répète. CBC* est parallélisable mais n'assure pas la sécurité sémantique.

Exercice 3: Attaque sur CBC avec le padding RFC2040

Rappels de cours. Block-Cipher:

- le message clair est découpé en blocs d'une taille fixée et chacun des blocs est chiffré.

- La longueur n des blocs et la taille l de la clef sont 2 caractéristiques des block-cipher.

- Le message *m* à chiffrer est découpé en blocs de n bits.
  - $m = m_1 m_2 ... m_k$

- Comment faire si la longueur du message *n'est pas un multiple de la longueur d'un bloc* ?

Exercice 3: Attaque sur CBC avec le padding RFC2040

➔ On le complète avec un padding.

- L'une des technique est la RFC 2040:
  - on complète le dernier bloc par autant d'octets que nécessaire.
  - chaque octet a pour valeur le nombre d'octets ajoutés.
    - Exemple: on veut des blocs de 8 octets et m = $o_1 o_2 o_3 o_4 o_5$
      - Combien manque t-il d'octet ? 3
      - Quelle valeur se verront-ils attribuer ? 3

O_1	O_2	O_3	O_4	O_5	03	03	03

  - Pour un algorithme de chiffrement qui opère sur des blocs de 128 bits (16 o), le bloc de clair $m_1 ... m_{12}$ sera transformé en: $m_1 ... m_{12} || 04040404$

On représente la valeur d'un octet avec 2 chiffres hexa:

00 = 0, 01 = 1, …, 0A = 10, …, 10 = 15, …, FF = 255.

Exercice 3: Attaque sur CBC avec le padding RFC2040

Considérons un attaquant qui a intercepté un chiffré C = (C1, C2, . . . , Cn) produit par un système de chiffrement à blocs en mode CBC avec le processus de bourrage RFC2040. On suppose aussi v connu.

1. Montrer que si l'attaquant dispose d'un oracle qui détermine si le message clair associé à un chiffré arbitraire est bien formé pour l'encodage RFC2040, alors il peut déterminer l'encodage effectivement utilisé pour le chiffré C.

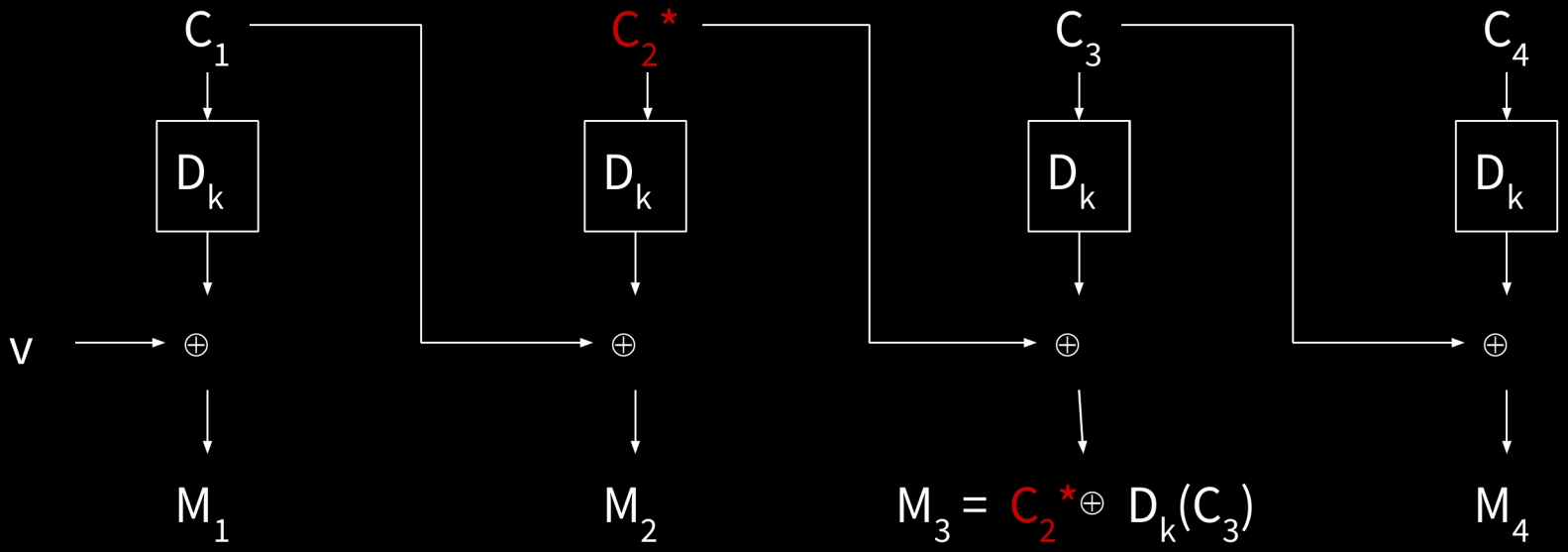
Déchiffrement de l'énoncé:

- Si on donne à l'oracle un chiffré qui donne un message clair mal encodé (selon RFC2040): l'oracle retourne une erreur.

- L'oracle vérifie donc si le clair associé à un chiffré se termine bien par `01`, ou `0202` ou `030303` ou `04040404`...

Exercice 3: Attaque sur CBC avec le padding RFC2040

Rappelons le déchiffrement CBC et intéressons nous à la propagation de l'erreur.

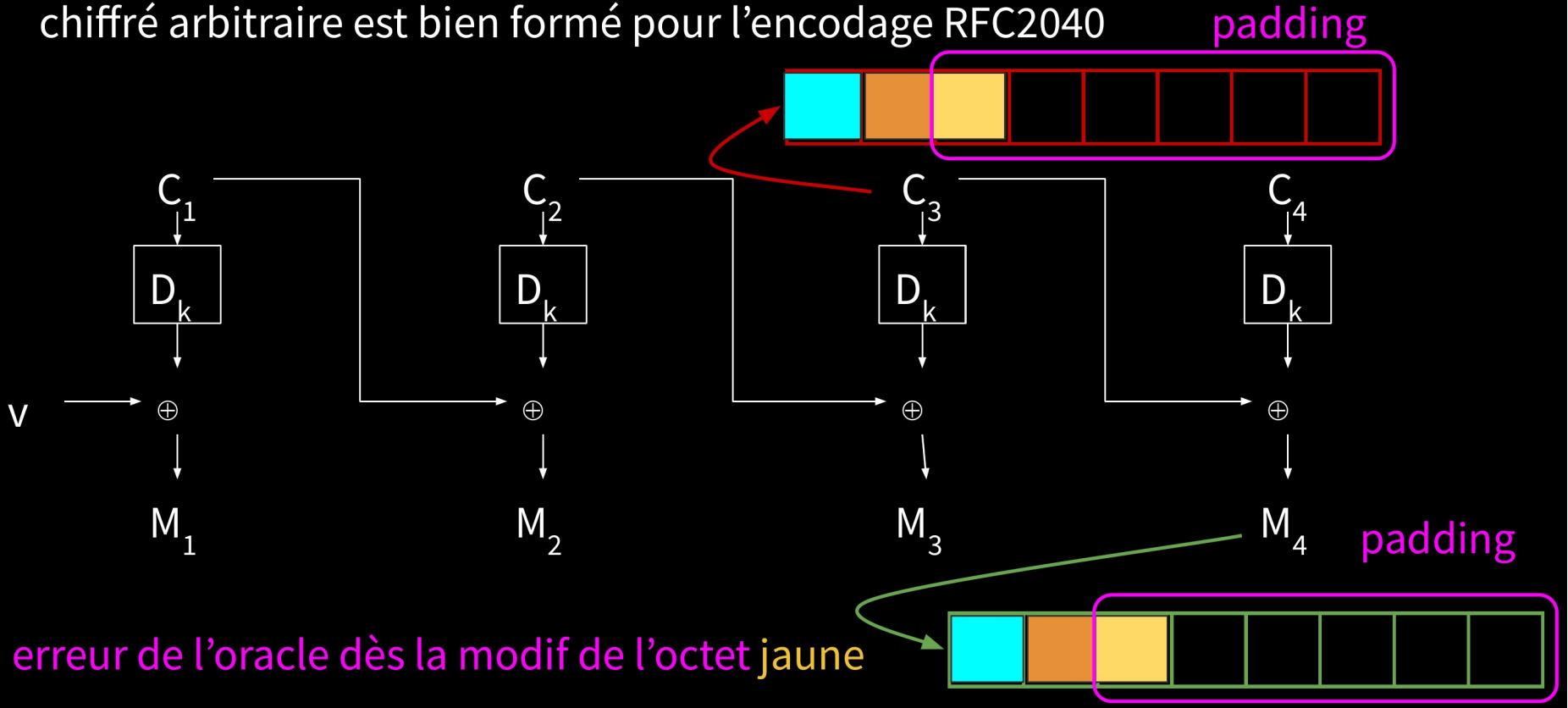


$M_2 = C_1 \oplus D_k(C_2^*)$
$M_3 = C_2^* \oplus D_k(C_3)$
$M_4 = C_3 \oplus D_k(C_4)$

➔ Si un bloc c_i dans un chiffré est modifié, alors le déchiffrement du bloc c_{i+1} donnera la même erreur en la même position

Exercice 3: Attaque sur CBC avec le padding RFC2040

- L'attaquant dispose d'un oracle qui détermine si le message clair associé à un chiffré arbitraire est bien formé pour l'encodage RFC2040



Exercice 3: Attaque sur CBC avec le padding RFC2040

➔ Pour déterminer l'encodage RFC2040:

- On produit une erreur sur le premier octet de l'avant-dernier bloc du chiffré, celle ci se propage sur le dernier bloc du clair. On interroge l'oracle.
- On fait une erreur sur le deuxième octet de l'avant-dernier bloc du chiffré et on interroge l'oracle.
- (...)
- Dès que l'erreur se trouve sur le premier octet de la fonction de padding l'encodage n'est plus correct et l'oracle de vérification le notifie.
- Une fois cette position connue, l'attaquant déduit la valeur de l'encodage aisément.
  - *Ex*: si on a des blocs de 16 et qu'une erreur apparaît à la douzième vérification, quelle est la forme du dernier bloc ?
    $M = m_1 ... m_{11} \,||\, \texttt{05 05 05 05 05}$

Exercice 3: Attaque sur CBC avec le padding RFC2040

2. Modifier l'attaque pour qu'il détermine le dernier octet du dernier bloc de clair.

- De la question précédente, on a:
  - Pour un bloc de chiffré en position C_i, un bloc de clair M_{i+1}:
    $M_{i+1} = C_i \oplus D_k(C_{i+1})$
  - Pour un bloc de chiffré *modifié* en position C_i^*, un bloc de clair M_{i+1}^*:
    $M_{i+1}^* = C_i^* \oplus D_k(C_{i+1})$
  - D'où:

$$M_{i+1} \oplus M_{i+1}^* = C_i \oplus D_k(C_{i+1}) \oplus C_i^* \oplus D_k(C_{i+1}) = C_i^* \oplus C_i$$

On peut se servir de cette propriété pour résoudre le problème!

Exercice 3: Attaque sur CBC avec le padding RFC2040

Soit M_i le dernier bloc de clair de la forme: $M = m_1 \ldots m_{11} \,||\, 05\ 05\ 05\ 05\ 05$
L'objectif est de connaître m_{11}

- Soit C_{i-1} l'avant dernier bloc associé à M_i. L'astuce est de modifier C_{i-1} en $C_{i-1}{}^*$ pour pouvoir obtenir $M_i{}^*$ de la forme: $M = m_1 \ldots 06 \,||\, 06\ 06\ 06\ 06\ 06$

  - Comment faire ? Avant modification: on connaît les octets chiffrés correspondants aux 5 derniers derniers clairs égaux à 05.

$M_i = m_1 \ldots m_{11} || 0505050505 = C_{i-1} \oplus D_k(C_i) = c_1 \ldots c_{11} || c_{12} \ldots c_{16} \oplus D_k(C_i)$

Pour le dernier octet:

On veut obtenir: $M_i = m_1 \ldots m_{11} || 0505050506$

Pour rappel: $M_{i+1} \oplus M_{i+1}{}^* = C_i{}^* \oplus C_i$ On note pour ce dernier octet: $M_{16} \oplus M_{16}{}^* = C_{15}{}^* \oplus C_{15}$

➔ $06 \oplus 05 = C_{15}{}^* \oplus C_{15}$ $\qquad C_{15}{}^* = C_{15} \oplus 06 \oplus 05$

Exercice 3: Attaque sur CBC avec le padding RFC2040

- On a donc réussi à obtenir C_{16}^* tel que $M_{16}^* = 06!$

➔ On fait la même chose pour les 4 octet restant jusqu'à obtenir la connaissance de $C_{-1}^* = c_1...c_{11}||c_{11}^*...c_{16}^*$ correspondant à $M^* = m_1...m_{11}||0606060606$

- On souhaite maintenant trouver c_{11}^* tel que $m_{11}^* = 06$ également.

➔ On attaque l'oracle en essayant toutes les valeurs de l'octet c_{11}^*

- Combien de requêtes doit-on faire à l'oracle au maximum ? 256

➔ Une unique requête de vérification sera considérée comme correcte: celle qui correspond à un bon encodage (c'est à dire avec les 6 derniers octets de valeur 06 pour le message clair): $m_{11}^* = 06.$ On a donc obtenu: c_{11}^*

- A notre disposition nous avons donc: c_{11}, c_{11}^* et m_{11}^*, avec $m_{11}^* = 06$

➔ $m_{11} \oplus m_{11}^* = C_{11}^* \oplus C_{11}$ $m_{11} = C_{11}^* \oplus C_{11} \oplus 06$

Exercice 3: Attaque sur CBC avec le padding RFC2040

3. En itérant le processus, montrer que l'attaquant peut obtenir ainsi le message clair M1, M2, . . . , Mn en intégralité.

- L'attaquant peut recommencer l'attaque octet par octet et trouver m_{10}, m_9… jusqu'à trouver le bloc complet.
- Il recommence ensuite avec le bloc de clair précédent, pour obtenir un encodage correct pour RFC2040…
- En moyenne, l'attaquant doit faire 128 requêtes à l'oracle de vérification pour déterminer un octet du message clair.