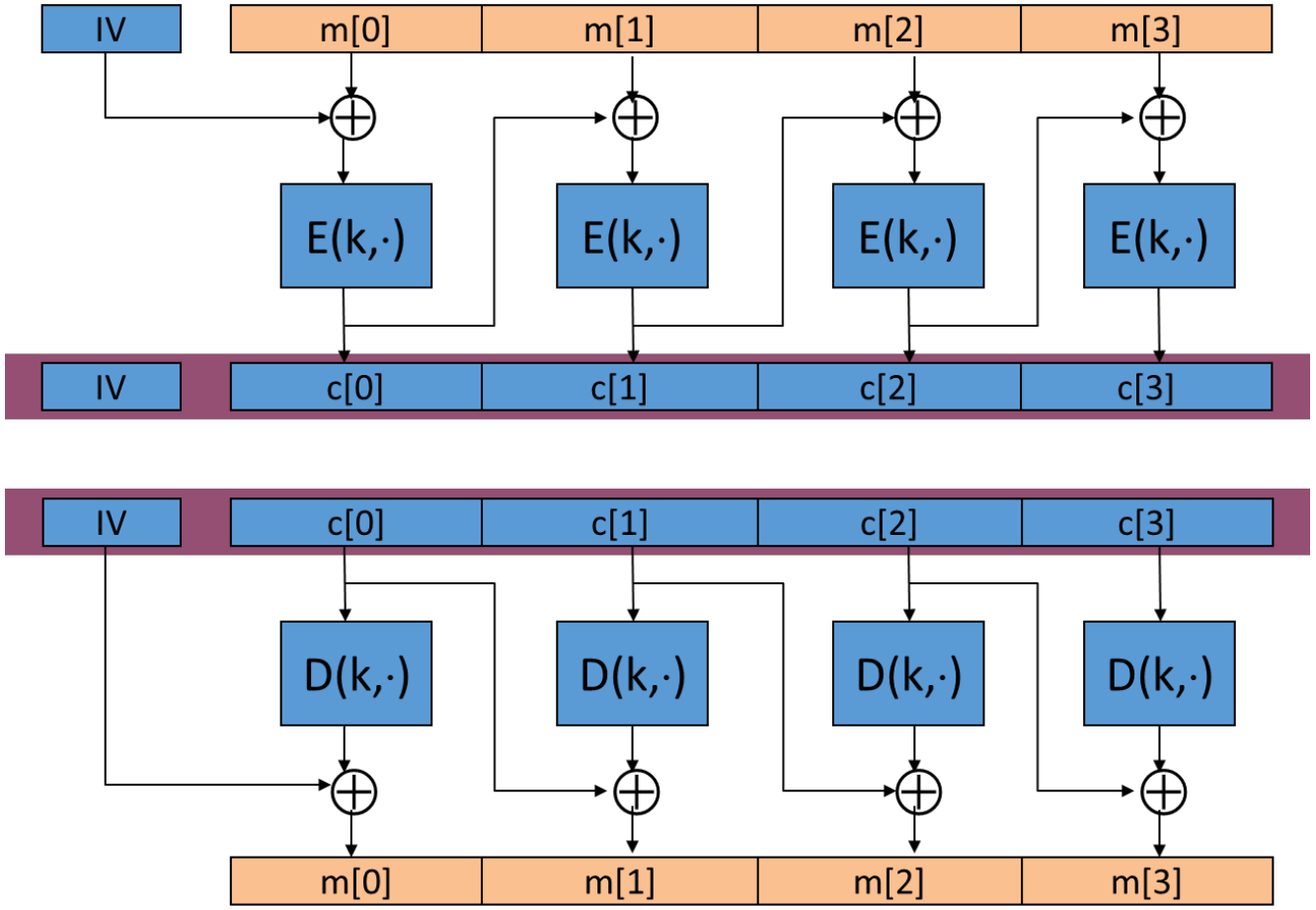
TD : Attaque sur l’oracle de padding

Padding = rembourrage

CBC : Chiffrement et déchiffrement



Rappel : Padding PKCS #5 / PKCS #7

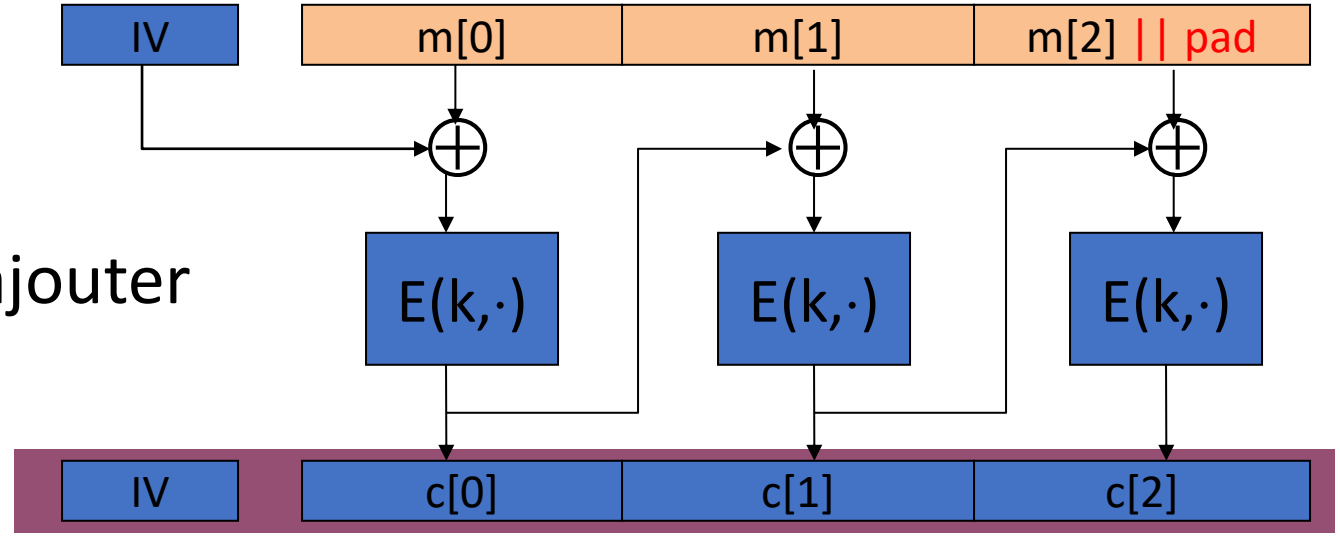
Dans le cas de messages dont la longueur n'est pas un multiple de la taille de bloc, il est nécessaire de compléter la taille pour le mode CBC

Ce pad est éliminé au déchiffrement.

PCKS7 : Pour un pad de n octets

n	n	n	•••	n

Si len(m) est un multiple, il faut ajouter un bloc entier.



Déchiffrement CBC

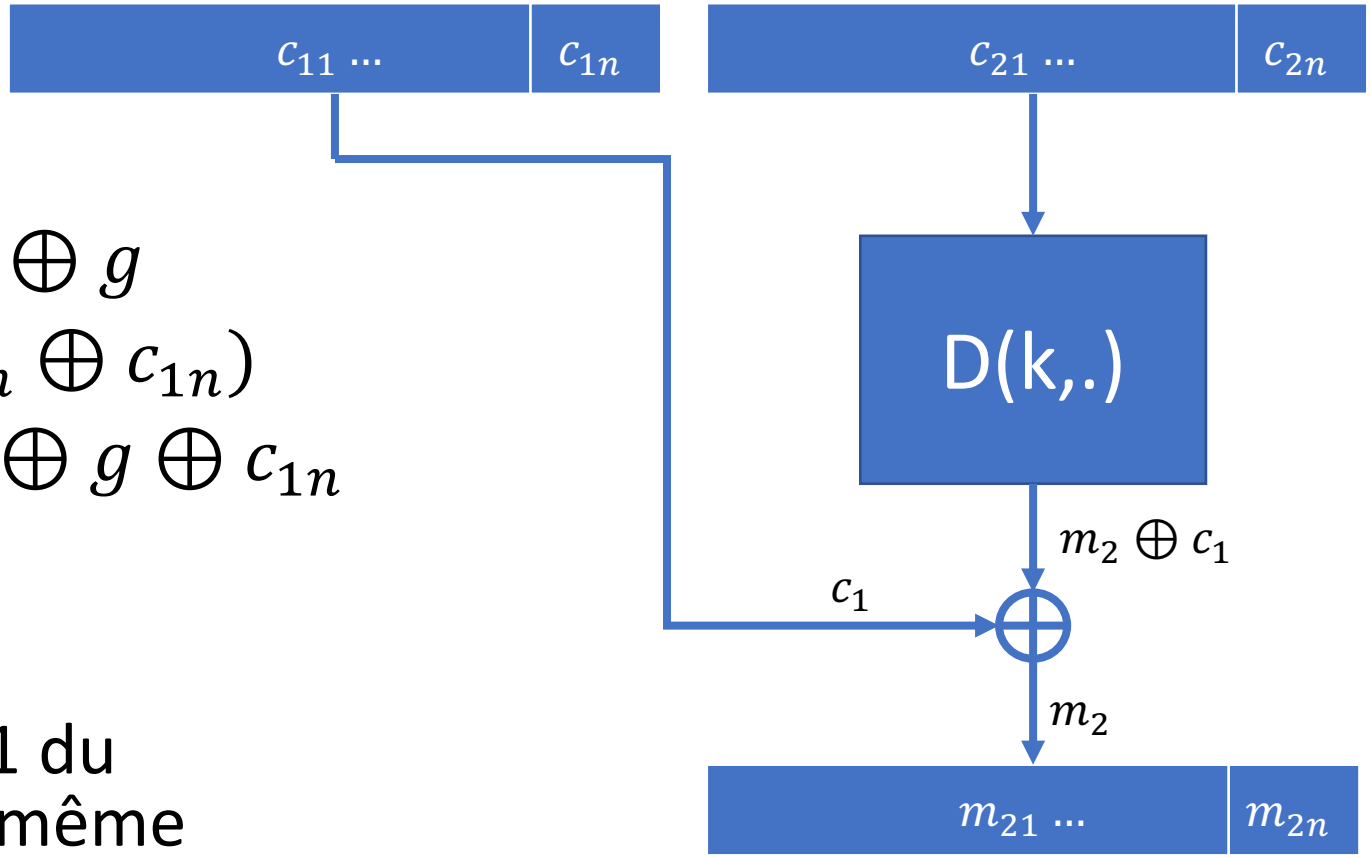
Soit un guess g

Remplaçons c_{1n} par $c'_{1n} = c_{1n} \oplus g$

On obtient $m'_{2n} = c'_{1n} \oplus (m_{2n} \oplus c_{1n})$

$= m_{2n} \oplus c_{1n} \oplus g \oplus c_{1n}$

$= m_{2n} \oplus g$



En modifiant un octet du bloc 1 du cryptogramme, on retrouve la même modification dans le bloc 2 du message

Attaque par oracle de padding

Si on arrive à construire un bloc se terminant par un padding légitime, le serveur accepte le déchiffrement du message, et on aura une erreur sur le contenu sémantique du message.

Si le padding est incorrect (par exemple $x0102$) : exception « Bad Padding Exception » (Java)

```
javax.crypto.BadPaddingException: Given final block not properly padded. Such issues
can arise if a bad key is used during decryption.
```

Si on trouve une valeur de g donnant un padding correct, on sait :

$$m'_{2n} = x01 = m_{2n} \oplus g \text{ et on en déduit}$$
$$\Rightarrow m_{2n} = g \oplus x01$$

Nous venons de découvrir le dernier octet du bloc du message.

Deuxième octet

Nous connaissons m_{2n} donc nous pouvons faire en sorte d'obtenir n'importe quelle valeur après déchiffrement, par exemple $x02$ en fixant

$$c'_{1n} = c_{1n} \oplus m_{2n} \oplus x02$$

Et nous allons donc modifier c_{1n-1} jusqu'à trouver $m'_{2n-1} = x02$

$m'_{2n} = x01 = m_{2n} \oplus g$ et on en déduit

$$\Rightarrow m_{2n} = g \oplus x01$$

Nous venons de découvrir le dernier octet du bloc du message.

Découvrir les autres octets…

$$m'_2 = m_{20} \| \cdots \| m_{2i-1} \| m_{2i} \oplus i \oplus g || i^{i-1}$$

- Si $m_{2i} \oplus i \oplus g = i,$ le padding est correct (et alors $m_{2i} = g$), le serveur accepte le message,
- Sinon, le padding est incorrect et le serveur refuse.

Il suffit donc de tester toutes les valeurs de g

$\boldsymbol{c'_1 = c_1 \oplus m^*_{2i} \oplus p_i \oplus g}$ et d'envoyer $\boldsymbol{c'_1 \| c_2}$ au serveur et analyser sa réponse

Raccourci – trouver la longueur du padding

- Le dernier bloc est normalement complété avec un padding.
- Il suffit de constater que si on modifie un octet du message, le padding reste valide, et si on modifie un octet du padding, le serveur considère le padding invalide.
- Donc pour le cryptogramme de l'avant dernier bloc, il suffit de tenter de modifier chaque octet l'un après l'autre, et analyser.

Pour le TD

- Serveur qui accepte une requête http, avec un paramètre passé en GET
- Forker le répository bitbucket :

  https://bitbucket.org/DamienSalvador/paddingoracleclient
- Modifier l'adrese IP/le port dans le fichier paddingOracleClient.java
- Vérifier que la connexion fonctionne (lancer le main, ou les unit tests)

- Le serveur répond 200 pour le message de départ, 403 en cas de message malformé (padding invalide) et 404 en cas de message incompris (padding valide)

TD … démarche conseillée

- *Il y a des tests, utilisez-les, complétez les !*

- Exécuter les tests jUnit pour voir ce qui marche
- Valider la connexion au serveur
- Remplir la fonction *splitMessageIntoBlocks()*
- Essayer de trouver le dernier octet, par exemple du 2eme bloc:
  - Fonction buildGuessForPosition en n'utilisant que les paramètres iv et guess
- Remplir la fonction *getPaddingArray()* pour construire un padding non chiffré
- Remplir la fonction *buildGuessForPosition()* pour trouver l'avant dernier octet, par exemple du 2eme bloc
- Puis généraliser
- Enlever le commentaire dans *runDecryptionForBlock() (pour décoder tous les blocs)*
- Remplir la fonction *getPaddingLengthForLastBlock()*
- Rentrer à la maison !

- *Il y a des tests, utilisez-les, complétez les !*

Complexité de l’attaque

- Si n’y a au maximum que 256 valeurs pour chaque octet.
- Pour un message de 40 octets, il faut donc 40x256=10240 essais, au lieu de (256)^40=2.10^96 …
- En pratique, on peut même réduire, en testant les octets par ordre de probabilité (Caractères « normaux » en premier).
- Sur TLS : Lucky 13, POODLE …