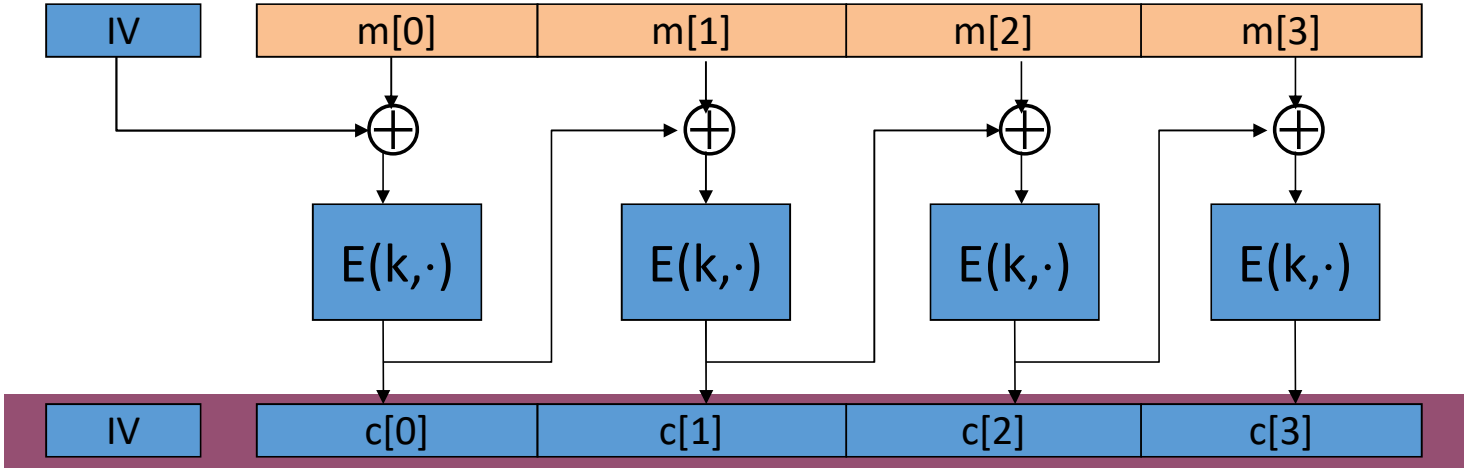
TD : Attaque sur l’oracle de padding

Padding = rembourrage

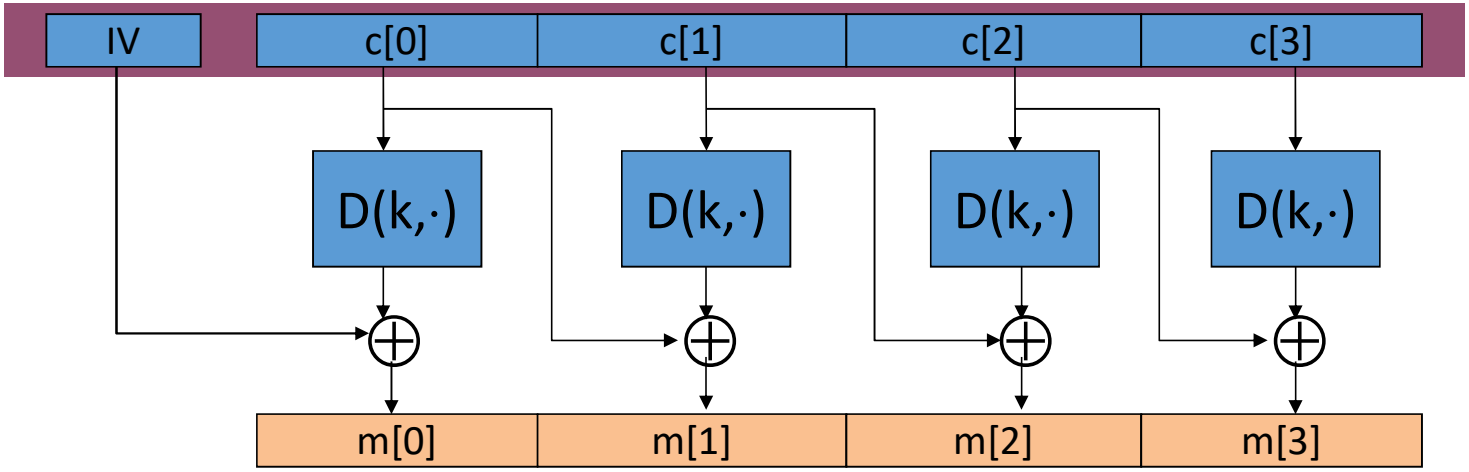
Rappel: CBC-Chain Block Cipher

On choisit IV au hasard



CBC : Déchiffrement

IV est récupéré du début du message



Déchiffrement CBC

Si $c_1' = c_1 \oplus (x00^{n-1}g)$

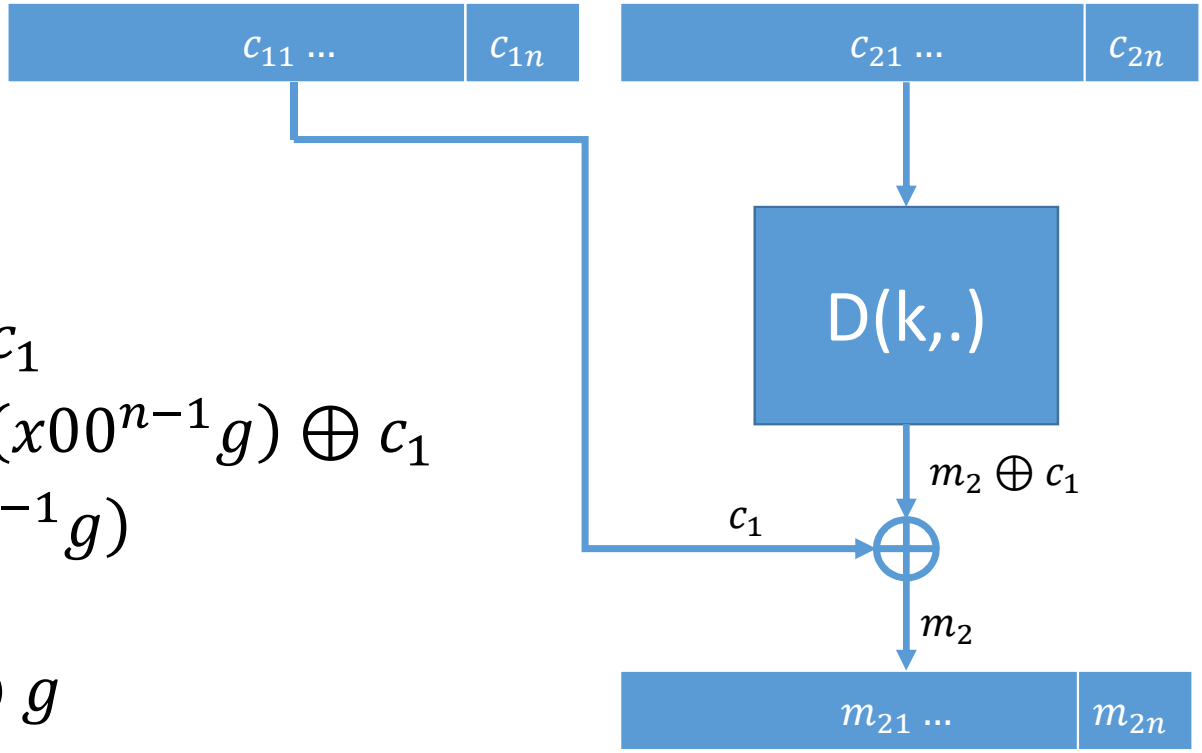
On obtient $m_2' = c_1' \oplus m_2 \oplus c_1$

$$m_2' = m_2 \oplus c_1 \oplus (x00^{n-1}g) \oplus c_1$$

$$m_2' = m_2 \oplus (x00^{n-1}g)$$

Et donc :

$$m_{2n}' = m_{2n} \oplus g$$



Rappel : Padding PKCS #5 / PKCS #7

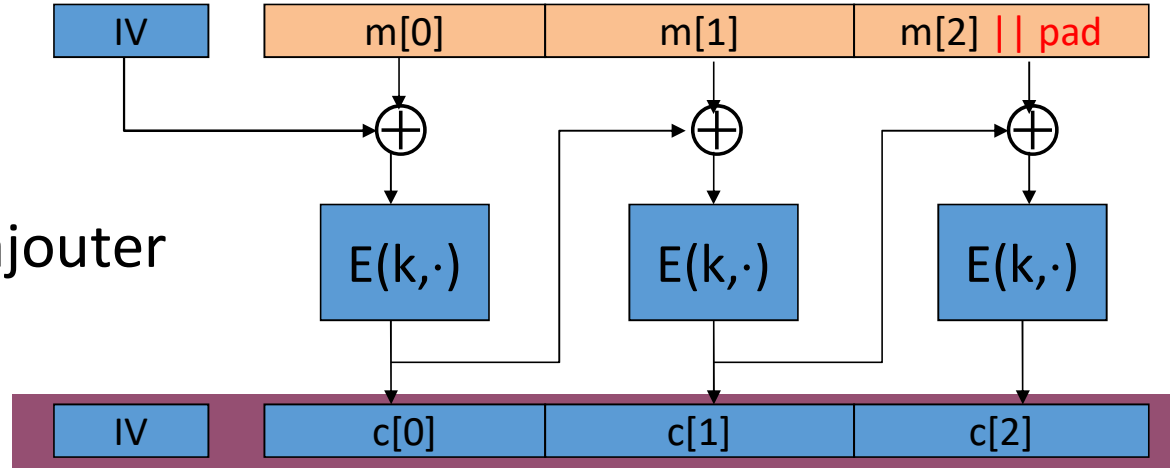
Dans le cas de messages dont la longueur n'est pas un multiple de la taille de bloc, il est nécessaire de compléter la taille pour le mode CBC

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.



Attaque par oracle de padding

Ainsi si on arrive à construire un bloc se terminant par un padding légitime, le serveur accepte le déchiffrement du message :

- Si $m'_{2n} = m_{2n} \oplus g \neq x01$: « Bad Padding Exception » (Java)
- Sinon, erreur sur le contenu sémantique du message.

- Si la proposition est bonne : $m'_{2n} = m_{2n} \oplus g = x01$ et on en déduit

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

Nous venons de découvrir le dernier octet du bloc.

Découvrir les autres octets

- Si l'on connait $m_{2,i+1}, \dots, m_{2,n-1}$ comment découvrir m_{2i}
- Notons $m^*_{2i} = x00^{n-i} \parallel x00 \parallel m_{2i+1} \parallel \cdots \parallel m_{2(n-1)}$
- Notons $p_i = x00^{n-i} \parallel i^i$ le masque correspondant au padding de longueur i
- Soit g un octet « tentative » (pour simplifier $x00^{n-i} \parallel g \parallel x00..$)

- Construisons :

$$\boldsymbol{c'_1 = c_1 \oplus m^*_{2i} \oplus p_i \oplus g = (c_{10} \dots c_{1(i-1)} \parallel c_{1i} \oplus i \oplus g \parallel c_{1(i+1)} \oplus m_{2(i+1)} \oplus i \parallel \cdots}$$

alors le message

$$m'_2 = m_2 \oplus c_1 \oplus c'_1$$
$$m'_2 = m_2 \oplus c_1 \oplus c_1 \oplus m^*_{2i} \oplus p_i \oplus g$$
$$m'_2 = m_2 \oplus m^*_{2i} \oplus p_i \oplus g$$
$$m'_2 = m_{20} \parallel \cdots \parallel m_{2i-1} \parallel m_{2i} \oplus i \oplus g || i^{i-1}$$

c	c1	c2	…	ci	c(i+1)	…	cn
m*	0	0	…	0	m(i+1)	…	mn
pi	0	0	…	i	i	…	i
g	0	0	…	g	0	…	0

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 $\boldsymbol{c'_1} \| \boldsymbol{c_2}$ 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
- Remplir la fonction *getPaddingArray()*
- Remplir la fonction *buildGuessForPosition()*
- Essayer de trouver l'avant dernier octet, par exemple du 2eme bloc
- Remplir la fonction *runDecryptionForBlock()*
- 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 …