Symmetric Key Encryption

Attacks on CBC

Levente Buttyán

CrySyS Lab, BME

buttyan@crysys.hu

Contents

1. Content leak problem
   – ciphertext-only or known-plaintext model
2. Padding oracle attack
   – adaptive chosen-ciphertext model
3. Exploiting predictable IVs
   – adaptive chosen-plaintext model

Content Leak Problem

Content leak problem

- let's assume that we have two encrypted blocks:

  $Y_i = E_K(X_i + Y_{i-1})$

  $Y_j = E_K(X_j + Y_{j-1})$

  that happen to be equal:

  $Y_i = Y_j$

- this means that

  $D_K(Y_i) = D_K(Y_j)$

  $X_i + Y_{i-1} = X_j + Y_{j-1}$

  $X_i + X_j = Y_{i-1} + Y_{j-1}$

- the attacker learns the difference $X_i + X_j$
- if X_i (or part of it) is known to the attacker, then X_j (or part of it) is also disclosed: $X_j = X_i + Y_{i-1} + Y_{j-1}$

Probability of a matching pair

- $\Pr\{ Y_i = Y_j \} = ?$

- assume that the block cipher works as a random function
- let P_k be the probability of having <u>no</u> matching pairs among k outputs (size of output space is $N = 2^n$)
  - $P_1 = 1$
  - $P_2 = ?$

$$\sum_{\text{for all } y} \Pr\{Y_1 = y\} \Pr\{Y_2 \neq y\} = N \frac{1}{N} \frac{N-1}{N} = \frac{N-1}{N}$$

Probability of a matching pair

- $\Pr\{ Y_i = Y_j \} = ?$

- assume that the block cipher works as a random function
- let P_k be the probability of having <u>no</u> matching pairs among k outputs (size of output space is $N = 2^n$)
  - $P_1 = 1$
  - $P_2 = \dfrac{N-1}{N}$
  - $P_3 = ?$

$$\sum_{\substack{\text{for all } y \\ \text{for all } y' \neq y}} \Pr\{Y_1 = y\} \Pr\{Y_2 = y'\} \Pr\{Y_3 \neq y, y'\}$$

$$= N\,(N-1)\,\frac{1}{N}\,\frac{1}{N}\,\frac{N-2}{N} = \frac{N-1}{N}\,\frac{N-2}{N}$$

Probability of a matching pair

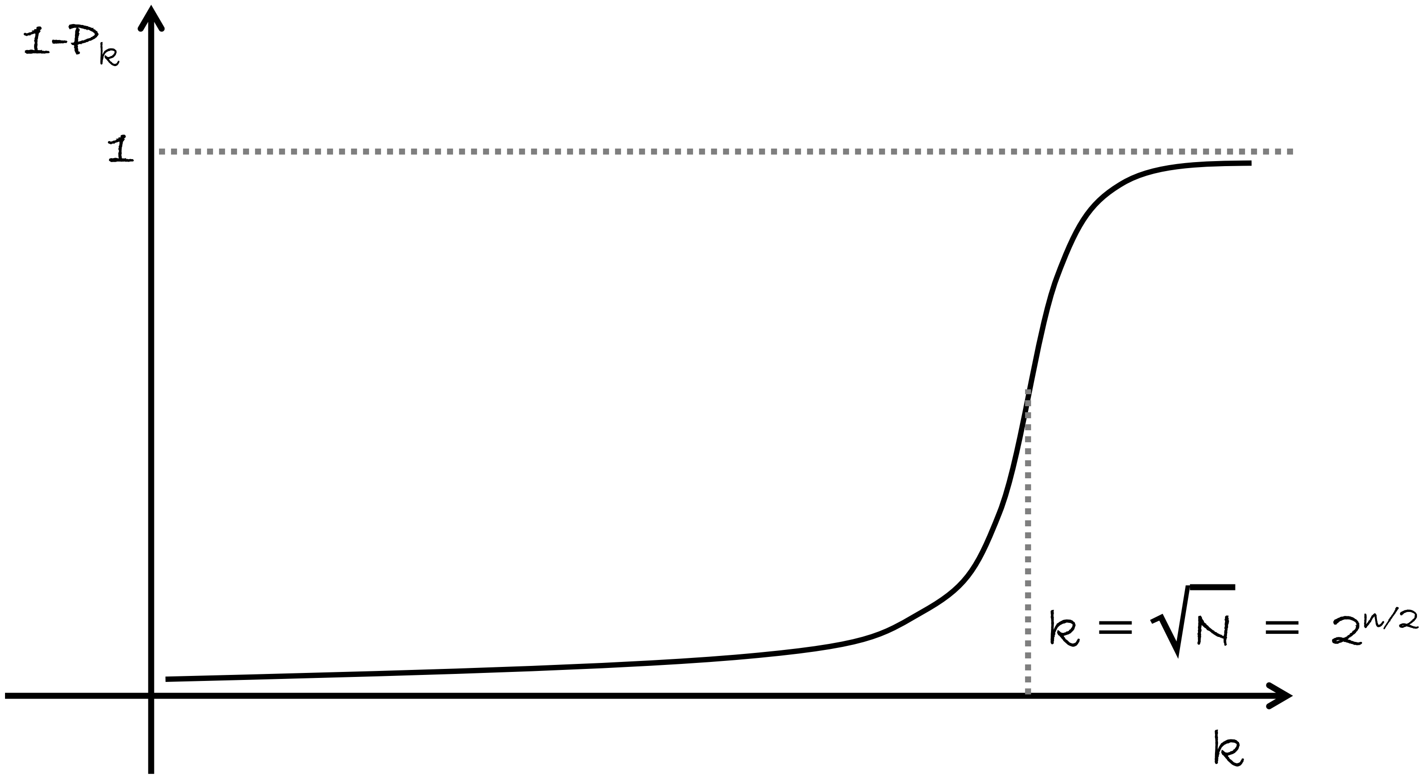
- $\Pr\{ Y_i = Y_j \} = ?$

- assume that the block cipher works as a random function
- let P_k be the probability of having <u>no</u> matching pairs among k outputs (size of output space is $N = 2^n$)
  - $P_1 = 1$
  - $P_2 = \dfrac{N-1}{N}$
  - $P_3 = \dfrac{N-1}{N} \dfrac{N-2}{N}$

    ...
  - $P_k = \dfrac{N-1}{N} \dfrac{N-2}{N} \cdots \dfrac{N-k+1}{N}$

Probability of a matching pair

- $\Pr\{ Y_i = Y_j \} = 1 - P_k$



Numerical example

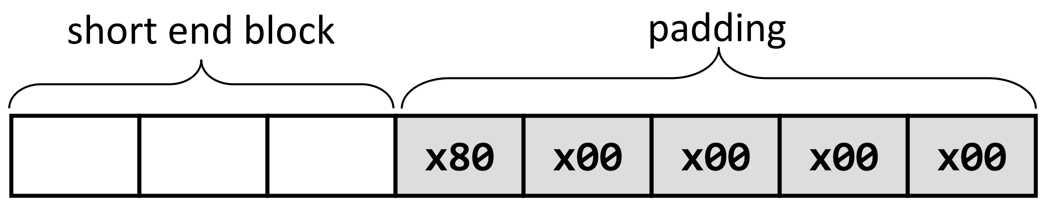
- let's assume that we use a block cipher with block length n = 64 bits (e.g., DES) in CBC mode
- among $2^{n/2} = 2^{32}$ encrypted blocks, there will be 2 identical blocks with large probability
  - information about the corresponding plaintext blocks is leaked
- as 1 block is 8 bytes (64 bits), 2^{32} blocks is just $8 \times 2^{32} = 2^{35}$ bytes, which is ~32GB
- it may be possible to observe that much encrypted data (e.g., an encrypted hard disk)

One should use a block cipher for which n/2 is sufficiently large, e.g., AES (n = 128 bits) or encrypt only small chunks of data with a given key.

Padding Oracle Attack

The padding oracle attack

- let’s assume two parties (e.g., a client and a server) communicate using a block cipher in CBC mode
- let’s assume they use the ISO 7816 padding scheme



- when the server receives any message (maybe from an attacker)...
  - it decrypts it according to the rules of CBC decryption
  - it tries to identify and remove the padding
- What should the server do, if the padding found is incorrect?
- if it sends a ”padding error” message, then it essentially leaks information...

The padding oracle attack

- Can we exploit this to decode something meaningful?

- an attack discovered by Serge Vaudenay in 2002 allows us to **decrypt any encrypted message efficiently** by repeatedly sending (adaptively) crafted ciphertexts to the server and observing its reponse
  - ”padding error” means that padding was not correct in what was obtained after decryption by the server
  - no error message means the padding was correct
  - we can play a ”yes/no questions” game with the server
  - if we ask cleverly, we can obtain all information we need!

- this is a special version of the (adaptive) **chosen ciphertext attack model**, where we choose a ciphertext, but we do not obtain the corresponding plaintext, only some partial information about the result of the decryption

The model

- let's assume we have an encrypted block $Y = E_K(X)$ and we don't know X and K
- we have access to an Oracle, which
  - knows and uses key K
  - decrypts whatever is sent to it using E in CBC mode with key K
  - checks the padding at the end of the decrypted input
  - tells whether the padding format was compliant with ISO 7816 (responds with PADDING_OK or PADDING_ERROR)
- we want to recover X

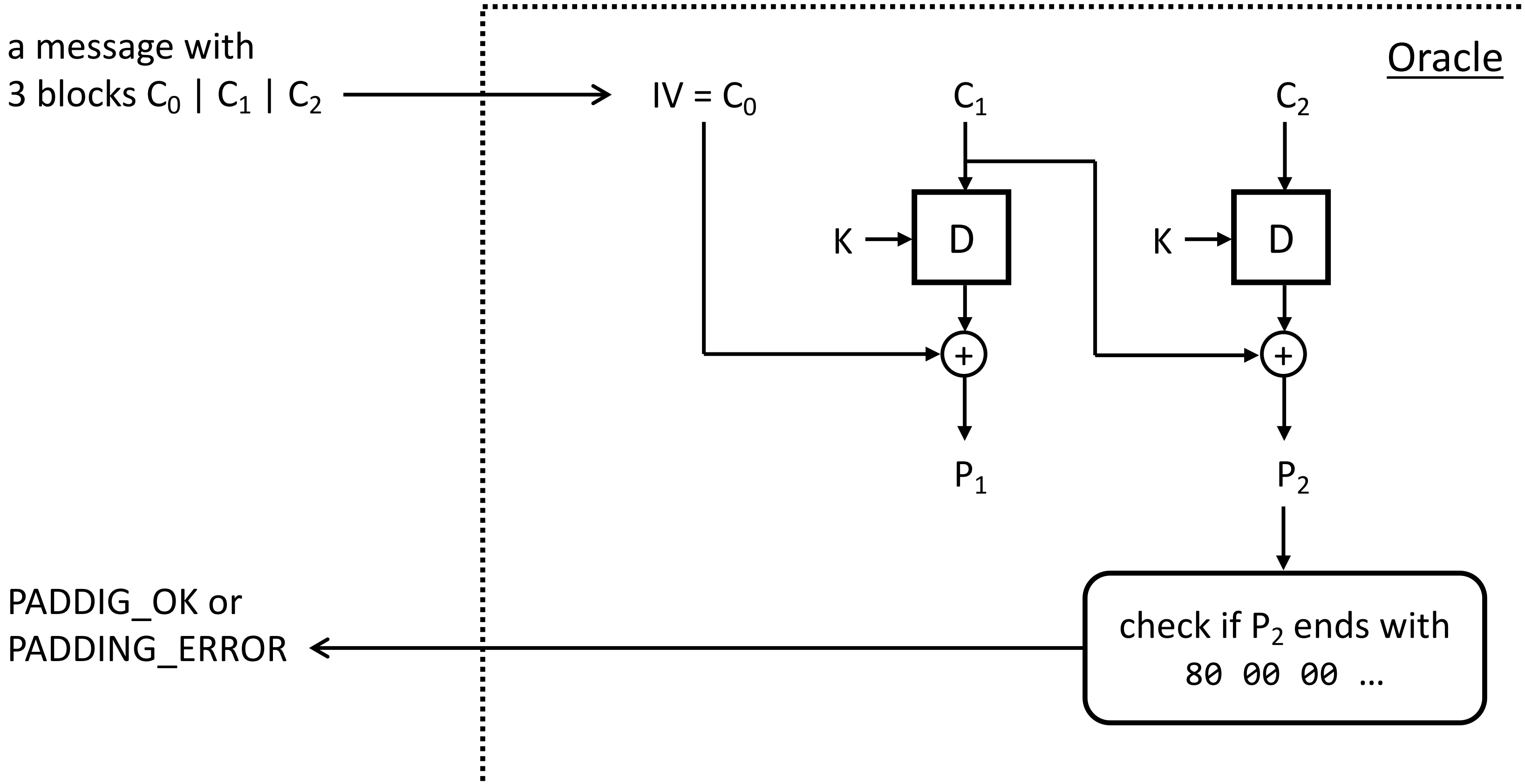
any input

Oracle

- decrypts the input using E in CBC mode with key K
- checks padding format ISO 7816
- responds with PADDING_OK or PADDING_ERROR

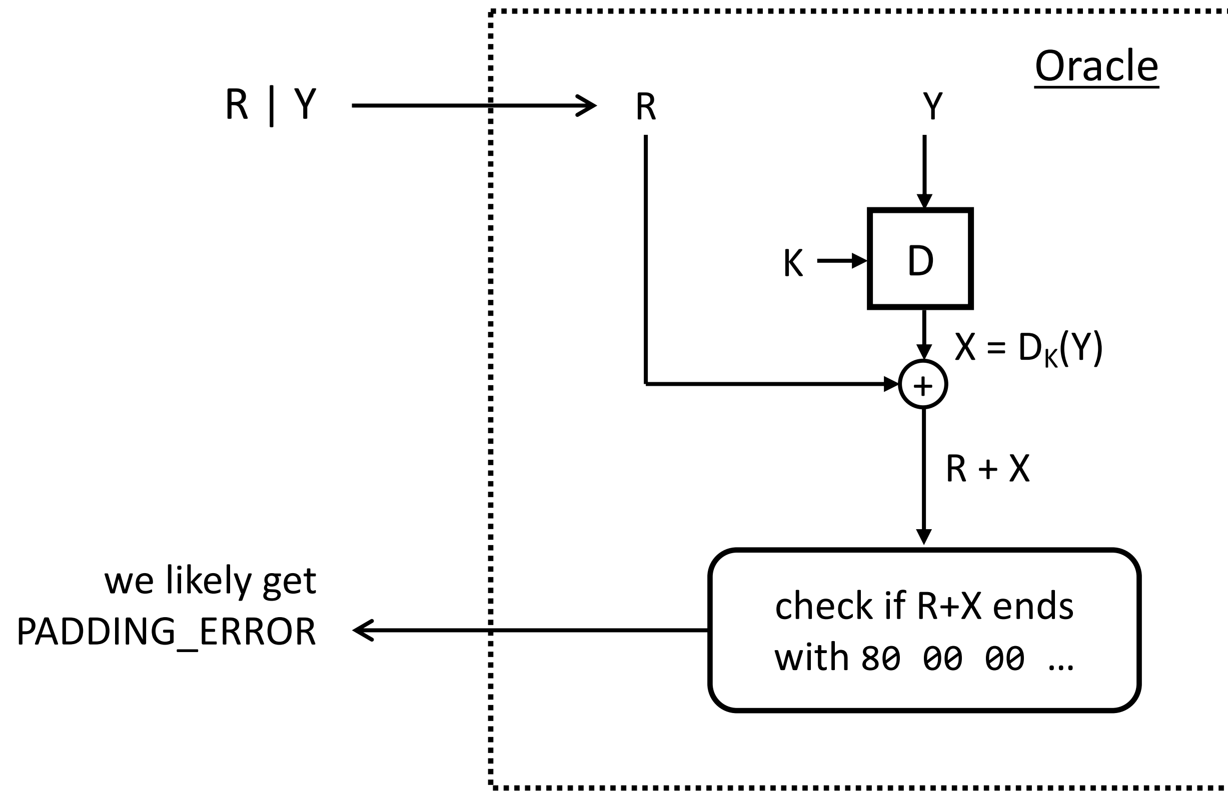
PADDING_OK / PADDING_ERROR

An example


a message with
3 blocks C0 | C1 | C2
Oracle
IV = C0
C1
C2
K
D
K
D
P1
P2
check if P2 ends with
80 00 00 …
PADDIG_OK or
PADDING_ERROR

The idea

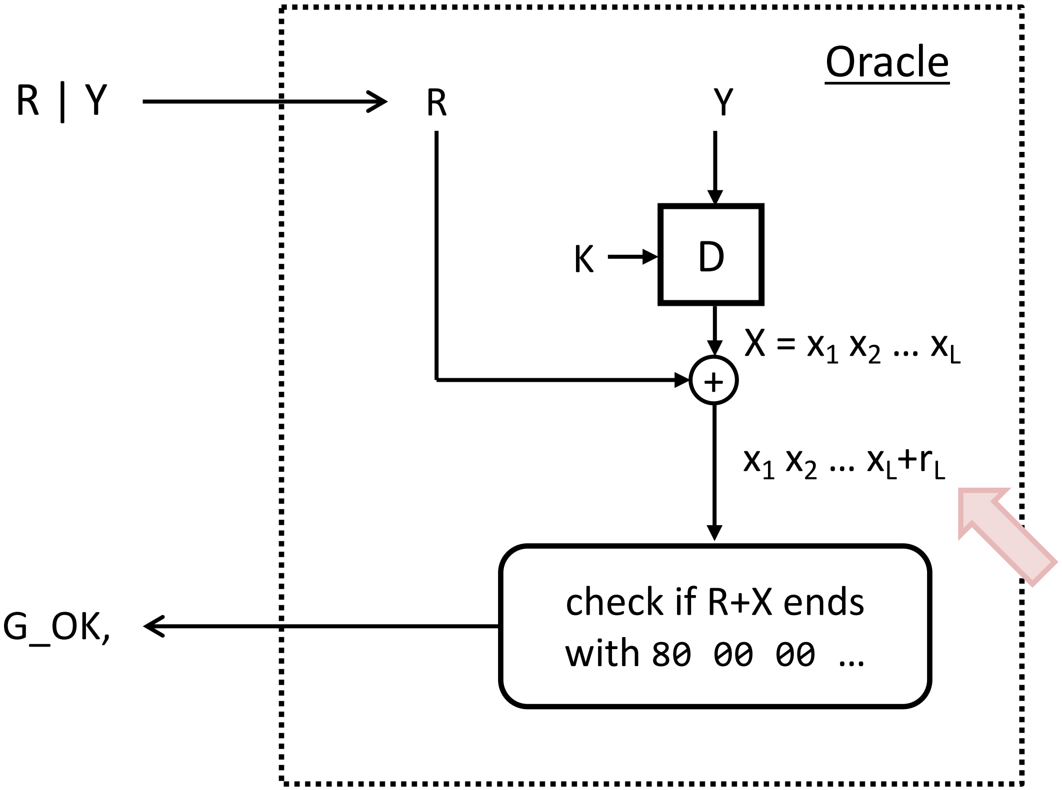
$Y = E_K(X)$ and
let $R =$ `00 00 … 00`



change (e.g., increment)
the last byte of R and repeat!

The idea

$Y = E_K(X)$ where $X = x_1\ x_2\ \ldots\ x_L$
and $R =$ `00 00` … $\mathbf{r_L}$

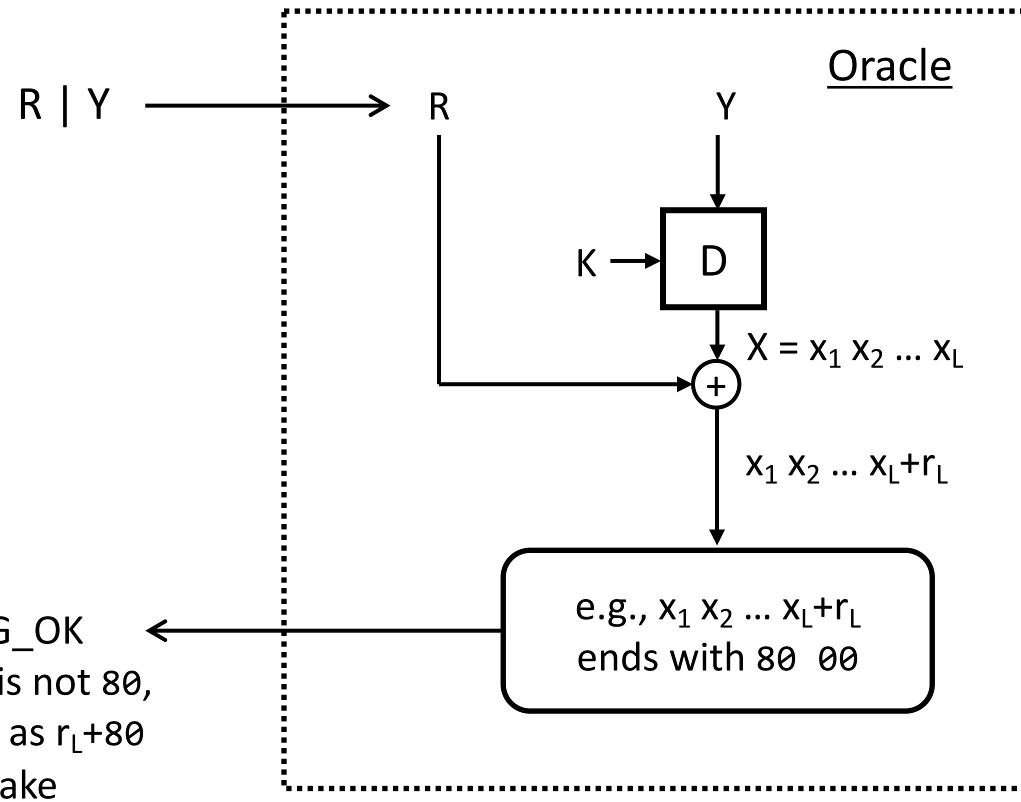


we eventually get PADDING_OK,
e.g., when $x_L + r_L =$ `80`

assuming that $x_L + r_L$ is indeed `80`,
we can compute $x_L = r_L$+`80`

But what if the padding was longer?

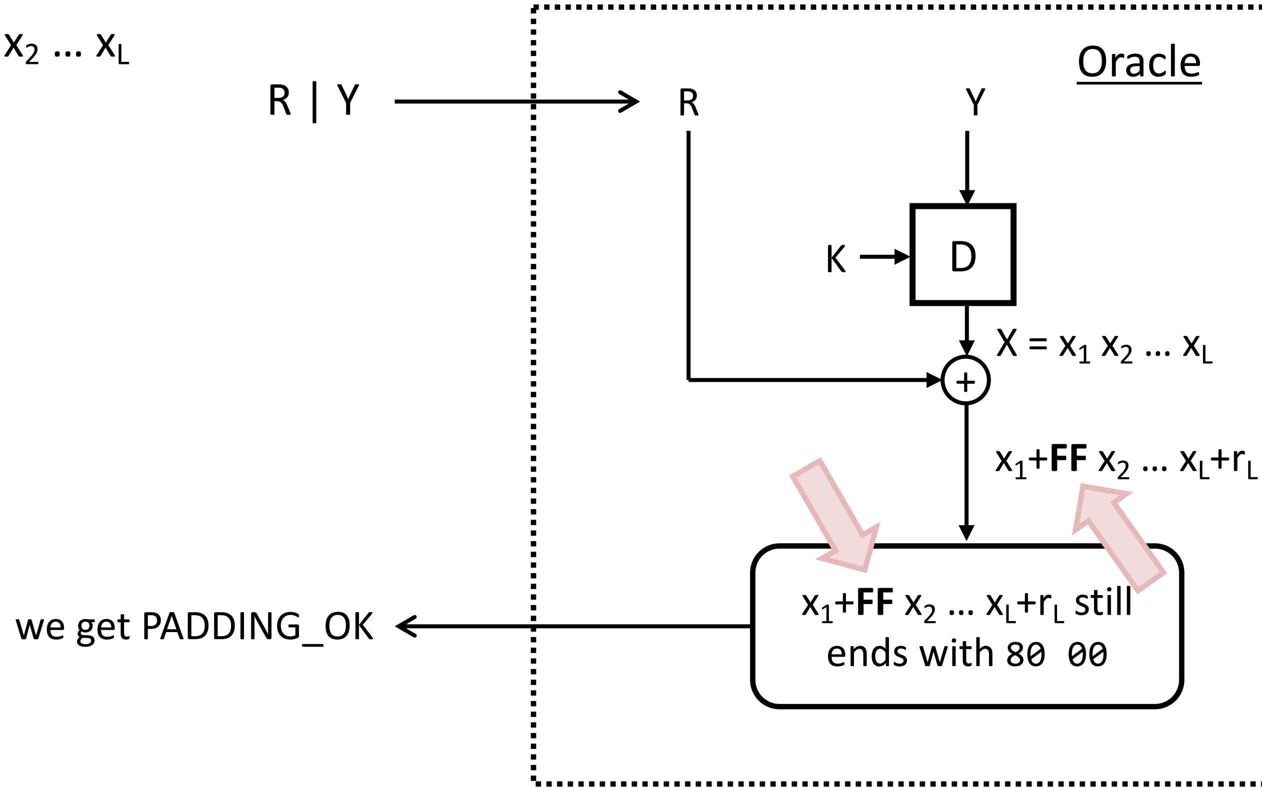
$Y = E_K(X)$ where $X = x_1\ x_2\ \ldots\ x_L$
and $R =$ `00 00` … r_L



we get PADDING_OK
although $x_L + r_L$ is not `80`,
so computing x_L as r_L+`80`
would be a mistake

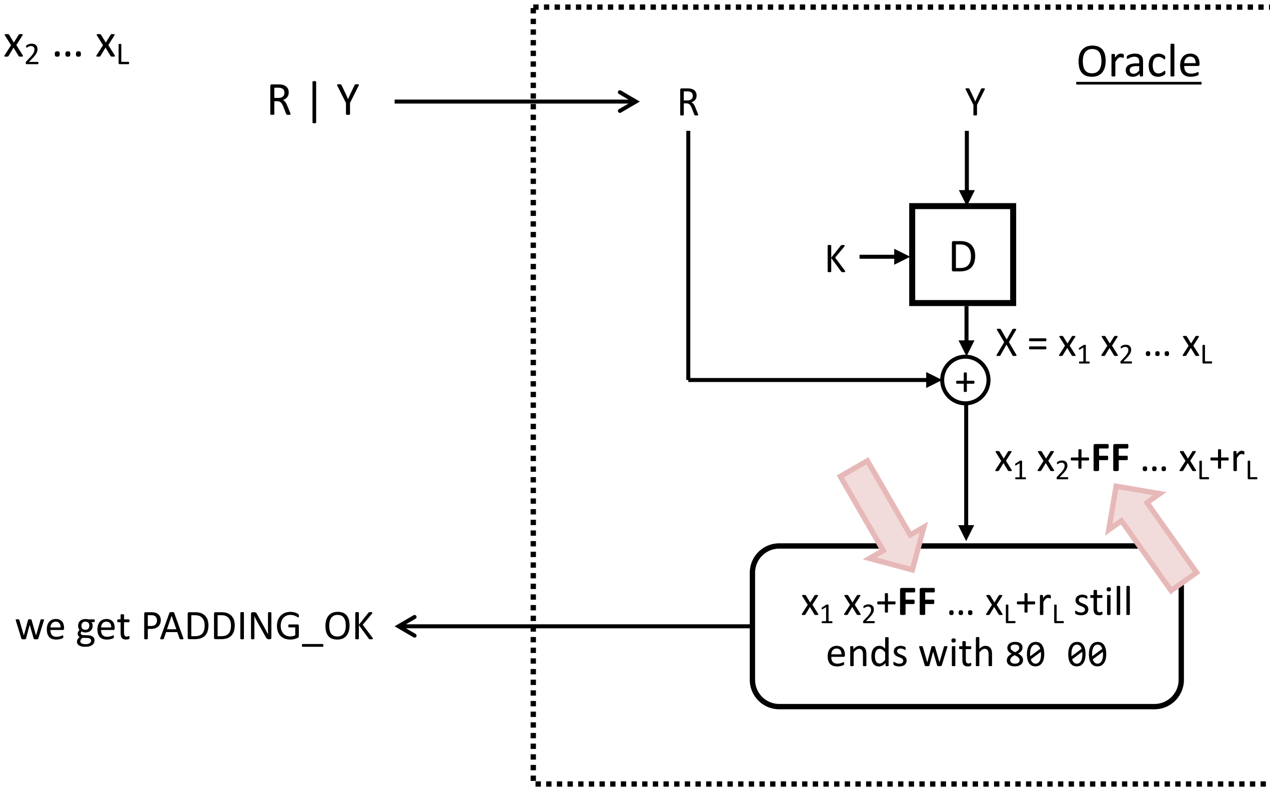
Another idea

$Y = E_K(X)$ where $X = x_1\ x_2\ \ldots\ x_L$
and $R =$ **`FF`** `00` … r_L



Another idea

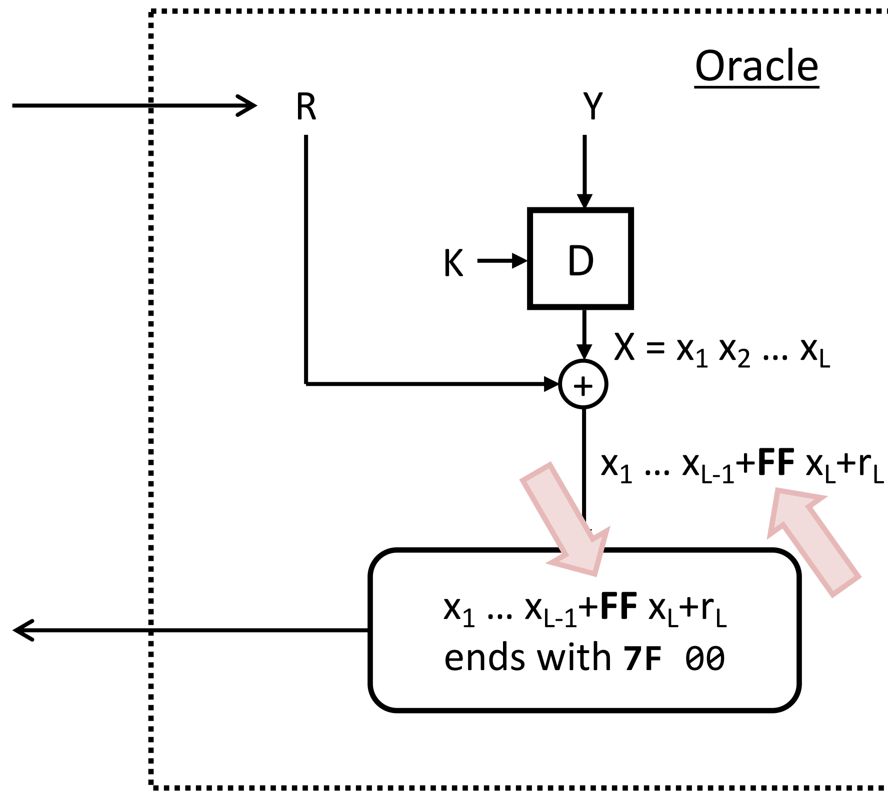
$Y = E_K(X)$ where $X = x_1\ x_2\ \dots\ x_L$
and $R =$ `00` **`FF`** … r_L



Another idea

$Y = E_K(X)$ where $X = x_1\, x_2 \ldots x_L$
and $R = $ `00` … **`FF`** r_L

R | Y



we get PADDING_ERROR,
and we learn the length
of the padding!

Another idea

knowing that for some $R = r_1\, r_2 \ldots r_L$ the padding length is plen, and hence, the padding is `80 00 … 00` (with length plen), we can compute

$x_{L\text{-plen}+1} = r_{L\text{-plen}+1} + 80$ and

$x_i = r_i + 00 = r_i$ for $i > L\text{-plen}+1$

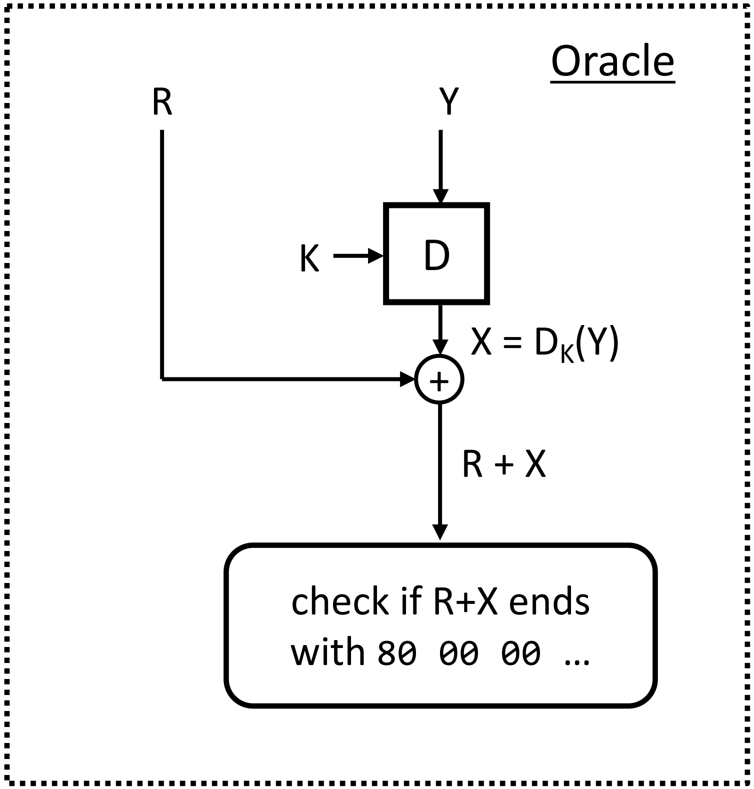
e.g., plen = 3 --» padding is `80 00 00`

$$
\begin{array}{cccc|c|c|c}
 & r_1 & r_2 & \ldots & r_{L-2} & r_{L-1} & r_L \\
+ & x_1 & x_2 & \ldots & x_{L-2} & x_{L-1} & x_L \\
\hline
 & . & . & . & 80 & 00 & 00
\end{array}
$$

$x_{L-2} = r_{L-2} + 80$

$x_{L-1} = r_{L-1}$

$x_L = r_L$



And the last step…

assume that for some $R = r_1\ r_2 \ldots r_L$ the padding length is plen, and hence, the padding is `80 00 … 00` (with length plen)

we can set $r_{L\text{-}plen+1} = x_{L\text{-}plen+1}$, which probably destroys the padding

but then we can change $r_{L\text{-}plen}$ until we get correct padding again, which means that the changed $r'_{L\text{-}plen} + x_{L\text{-}plen}$ must be `80`, and hence $x_{L\text{-}plen} = r'_{L\text{-}plen} + 80$

e.g., plen = 3 --» padding is `80 00 00`

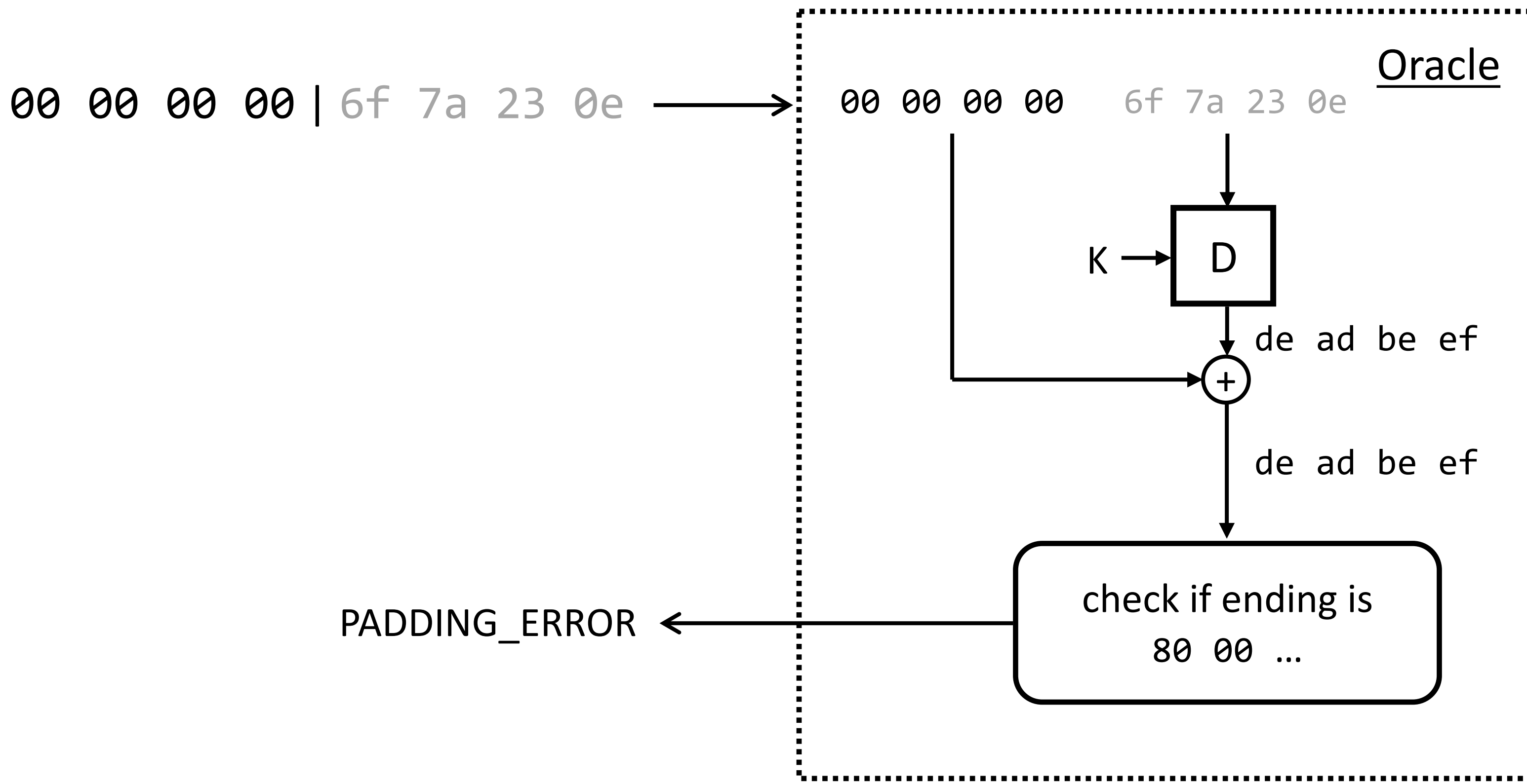
$$
\begin{array}{cccccc}
 & r_1 & r_2 & \ldots & \boxed{r_{L-2}} & \boxed{r_{L-1}} & \boxed{r_L} \\
+ & x_1 & x_2 & \ldots & x_{L-2} & x_{L-1} & x_L \\
\hline
 & . & . & . & 80 & 00 & 00
\end{array}
\qquad \Rightarrow \qquad
\begin{array}{cccccccc}
 & r_1 & r_2 & \ldots & \boxed{r'_{L-3}} & \boxed{x_{L-2}} & \boxed{x_{L-1}} & \boxed{x_L} \\
+ & x_1 & x_2 & \ldots & x_{L-3} & x_{L-2} & x_{L-1} & x_L \\
\hline
 & . & . & . & 80 & 00 & 00 & 00
\end{array}
$$

--» we can compute $x_{L-2}\ x_{L-1}\ x_L$

--» we can compute $x_{L-3} = r'_{L-3} + 80$

Example

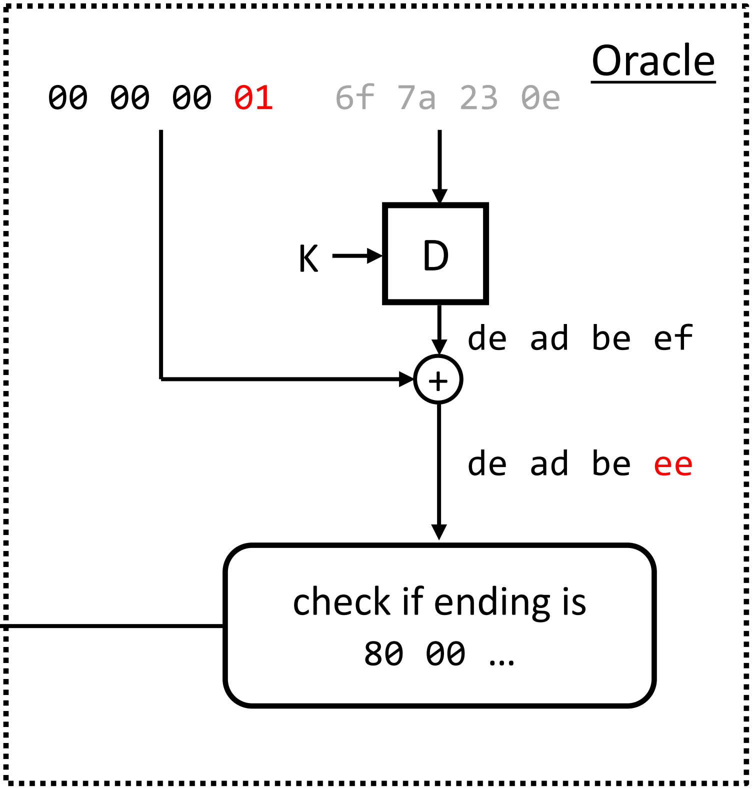
X = de ad be ef
Y = 6f 7a 23 0e
R = 00 00 00 00



Example

X = de ad be ef
Y = 6f 7a 23 0e
R = 00 00 00 01

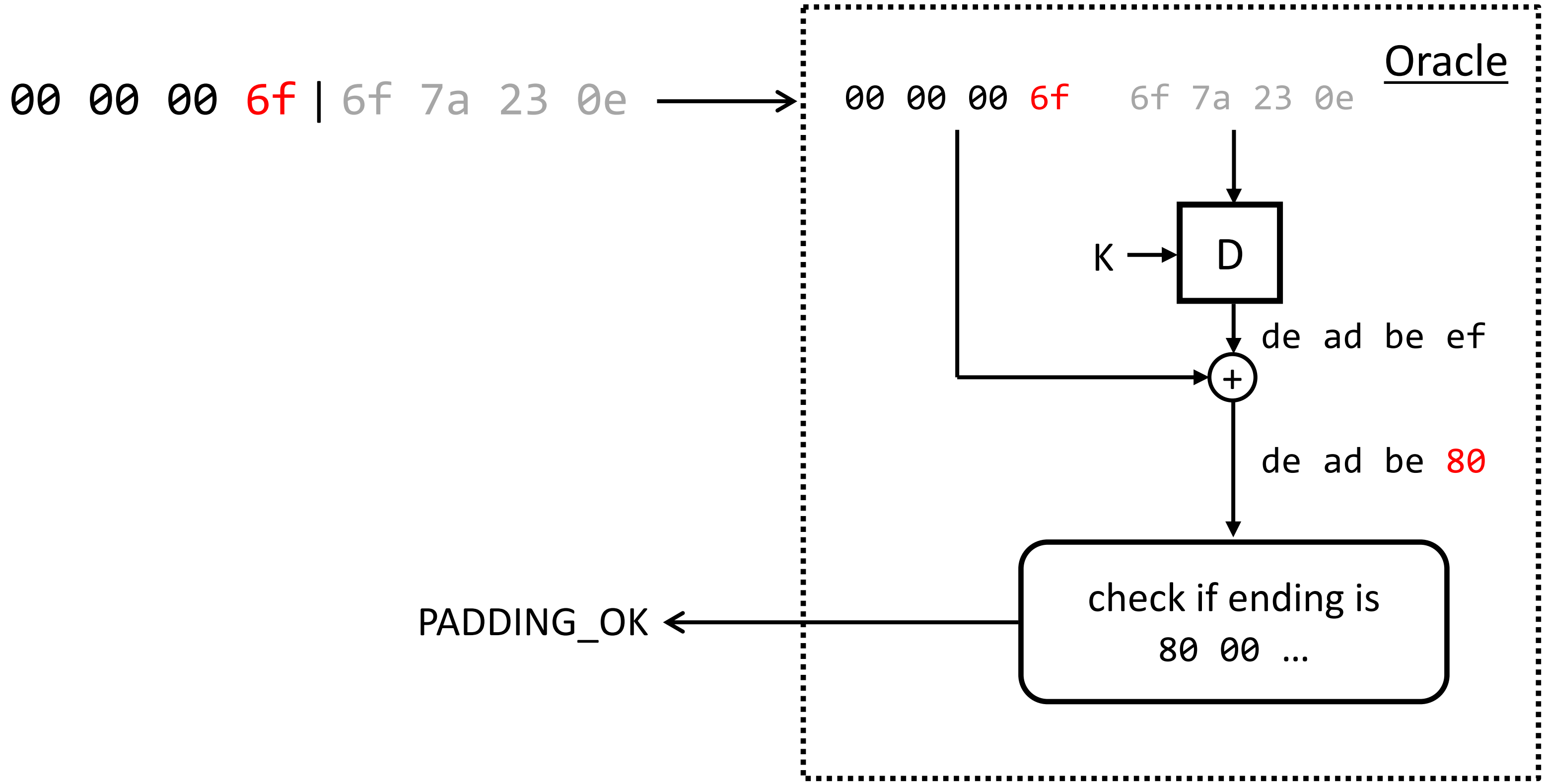
00 00 00 01 | 6f 7a 23 0e



PADDING_ERROR

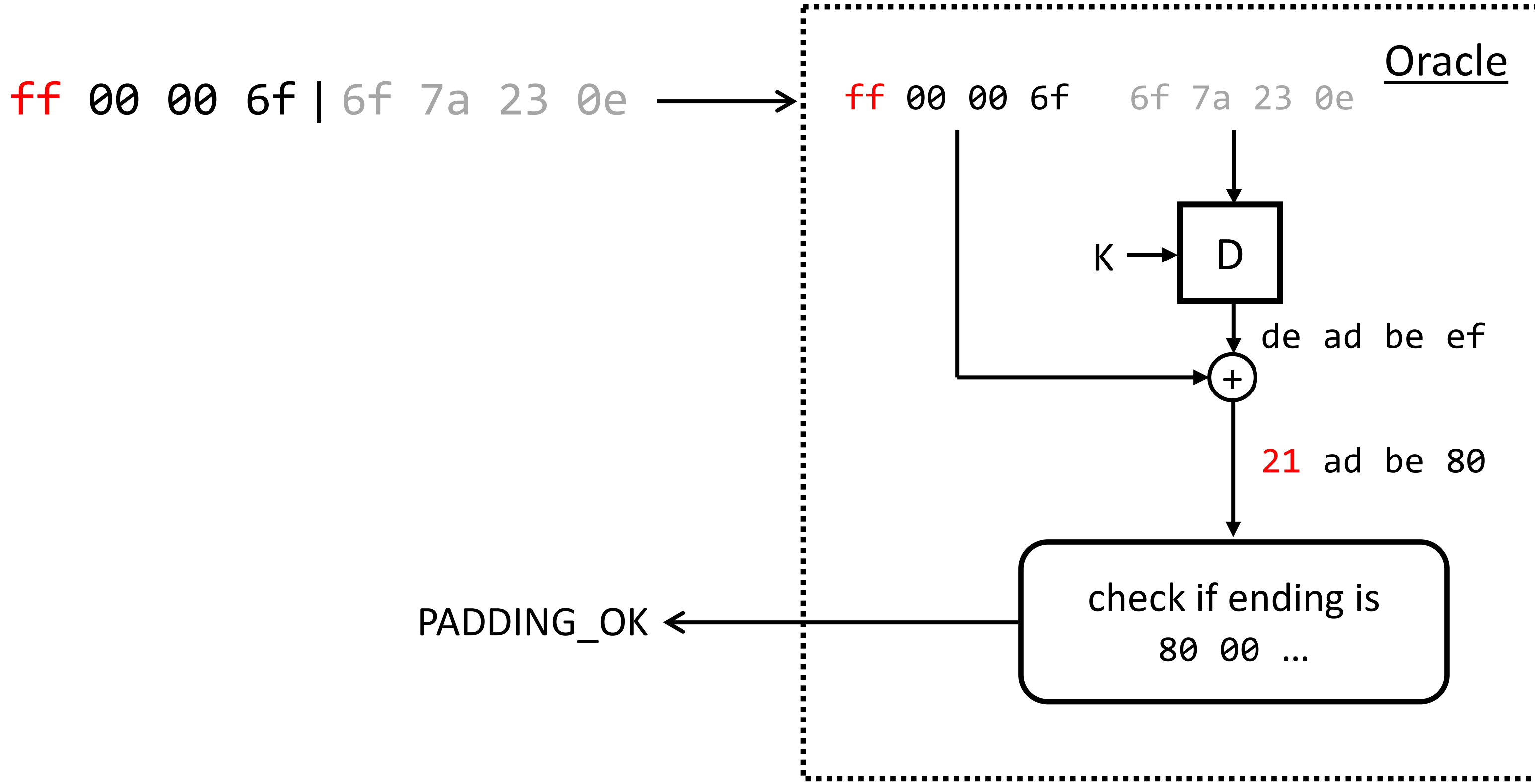
Example

```
X = de ad be ef
Y = 6f 7a 23 0e
R = 00 00 00 6f
```



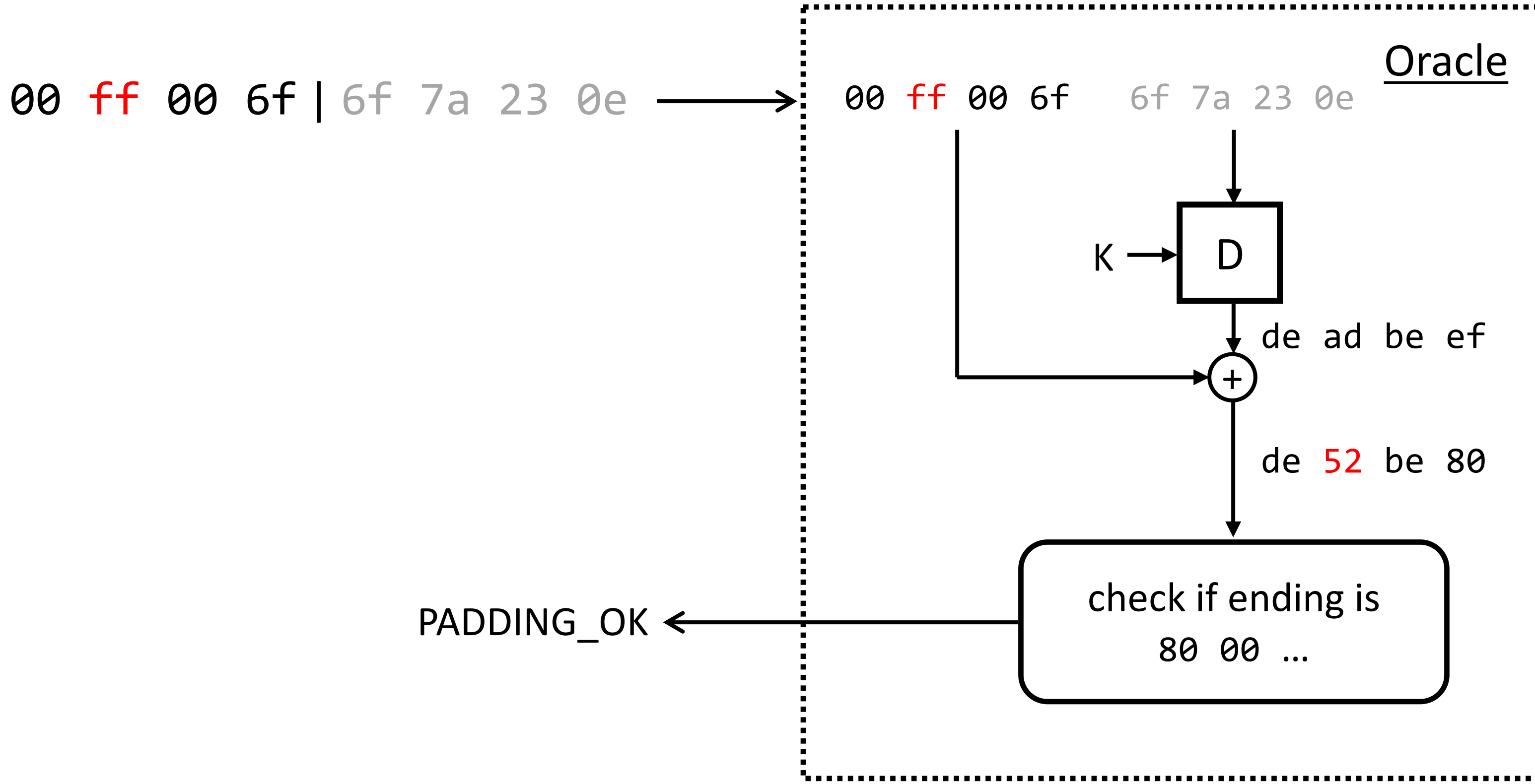
Example

```
X = de ad be ef
Y = 6f 7a 23 0e
R = ff 00 00 6f
```



Example

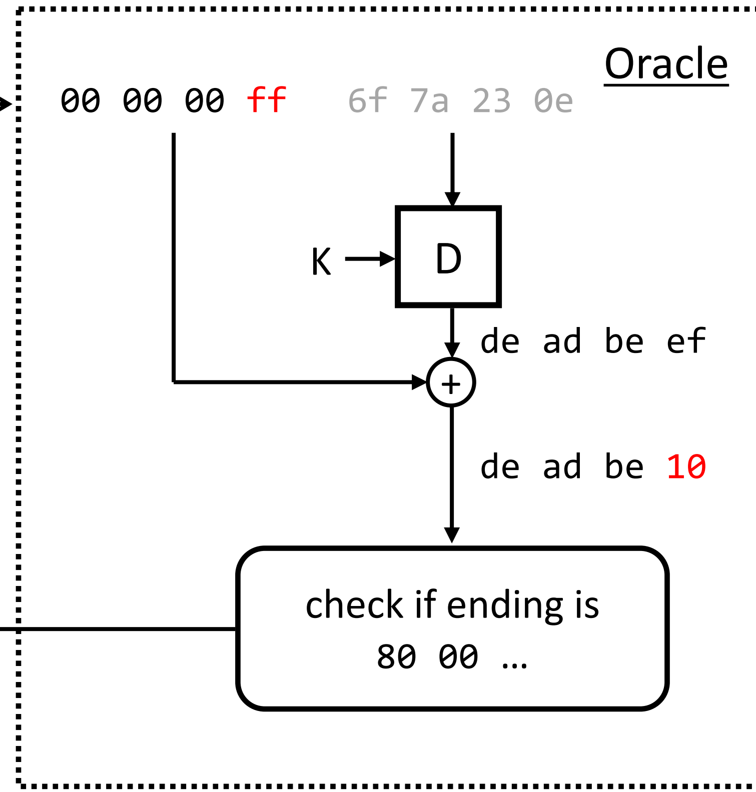
```
X = de ad be ef
Y = 6f 7a 23 0e
R = 00 ff 00 6f
```



Example

X = de ad be ef
Y = 6f 7a 23 0e
R = 00 00 00 ff

00 00 00 ff | 6f 7a 23 0e →



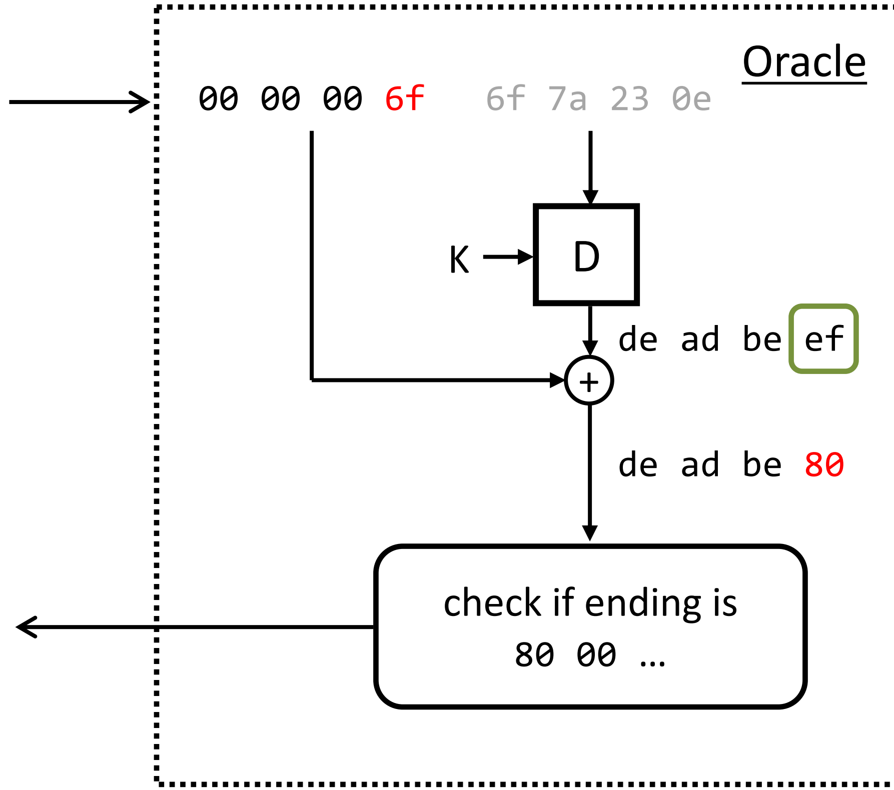
PADDING_ERROR ←

and we learn that the padding length is 1

Example

```
X = de ad be ef
Y = 6f 7a 23 0e
R = 00 00 00 6f
```

00 00 00 6f | 6f 7a 23 0e



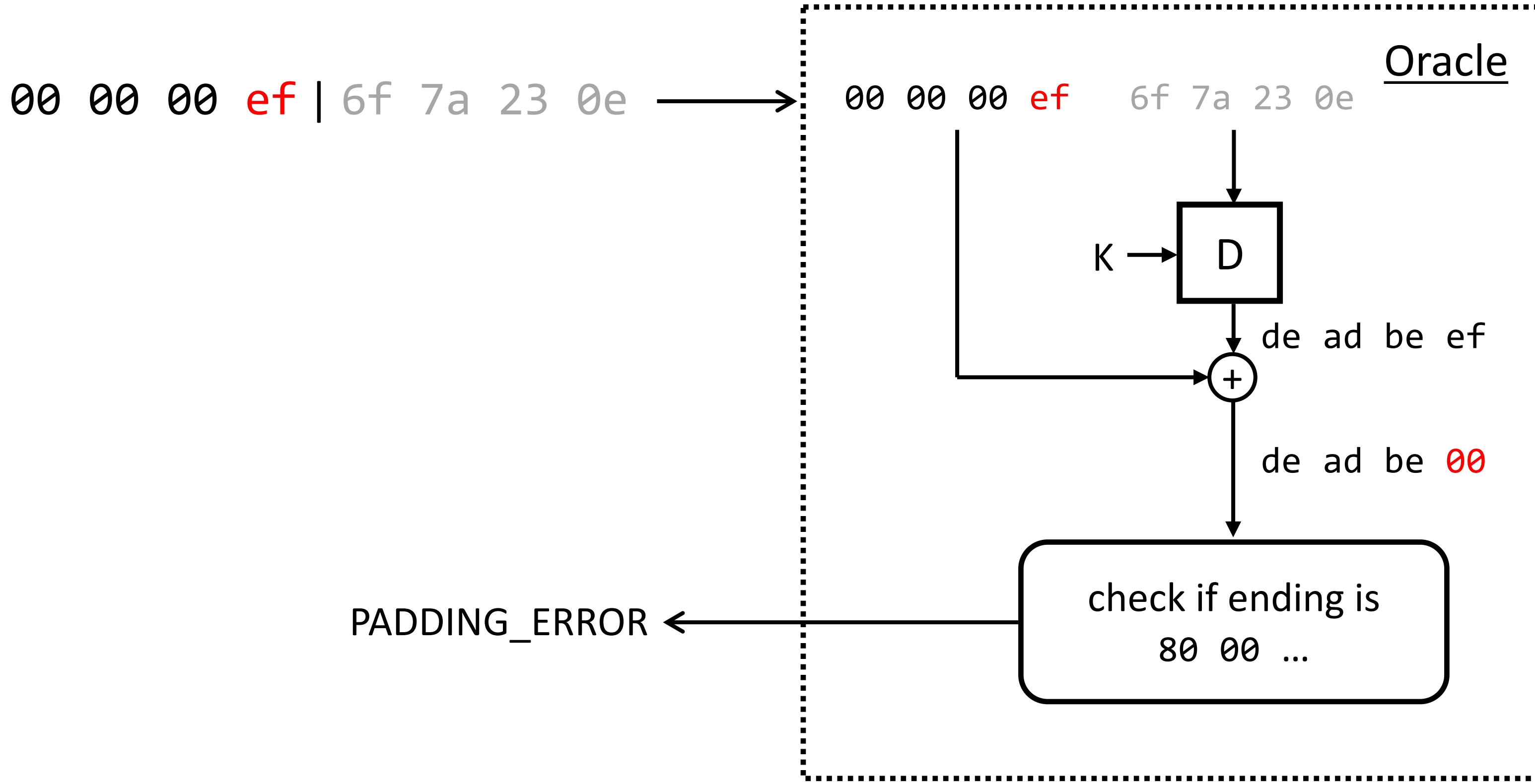
PADDING_OK

$6f + x_4 = 80$

$x_4 = 6f + 80 = \boxed{ef}$

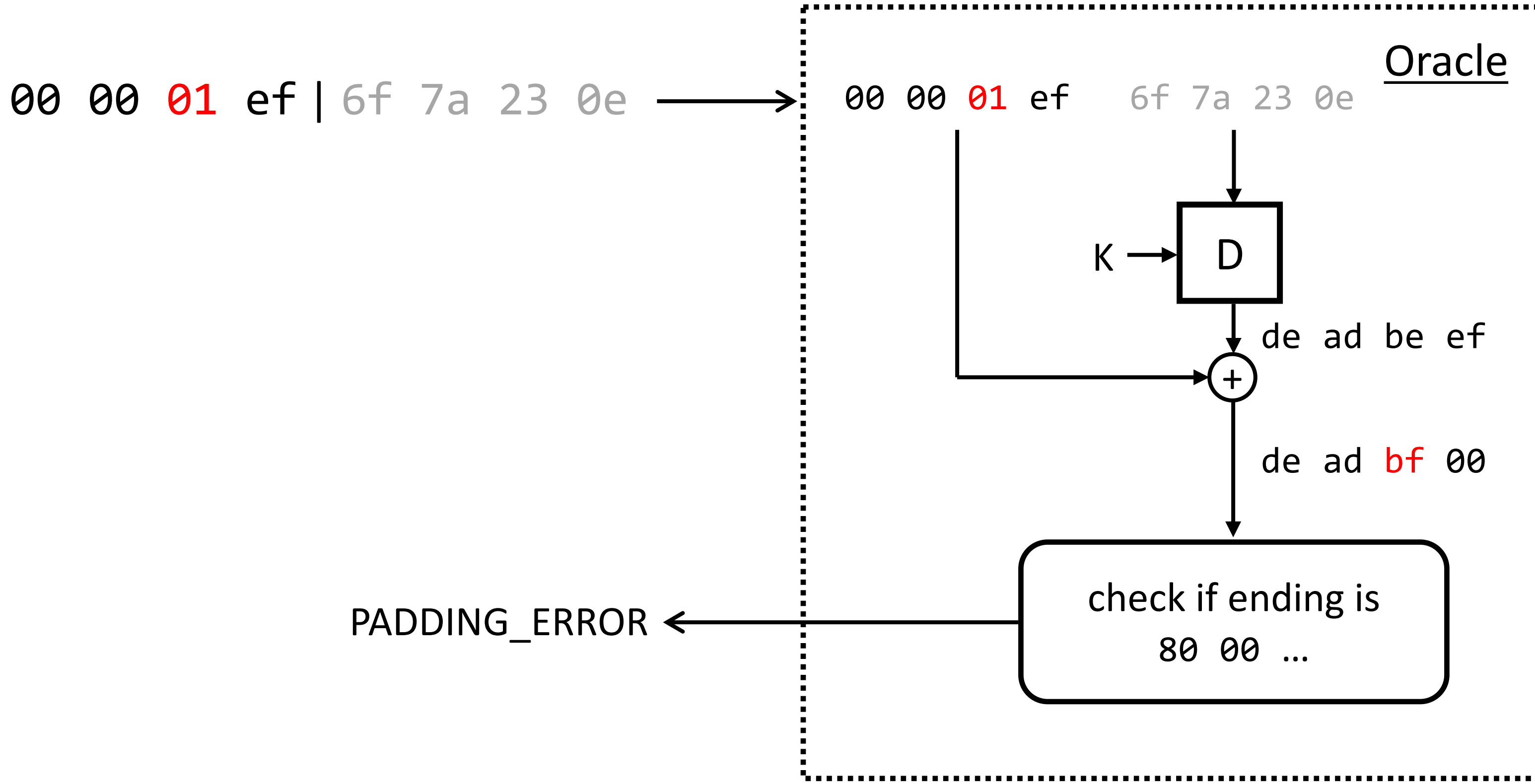
Example

```
X = de ad be ef
Y = 6f 7a 23 0e
R = 00 00 00 ef
```



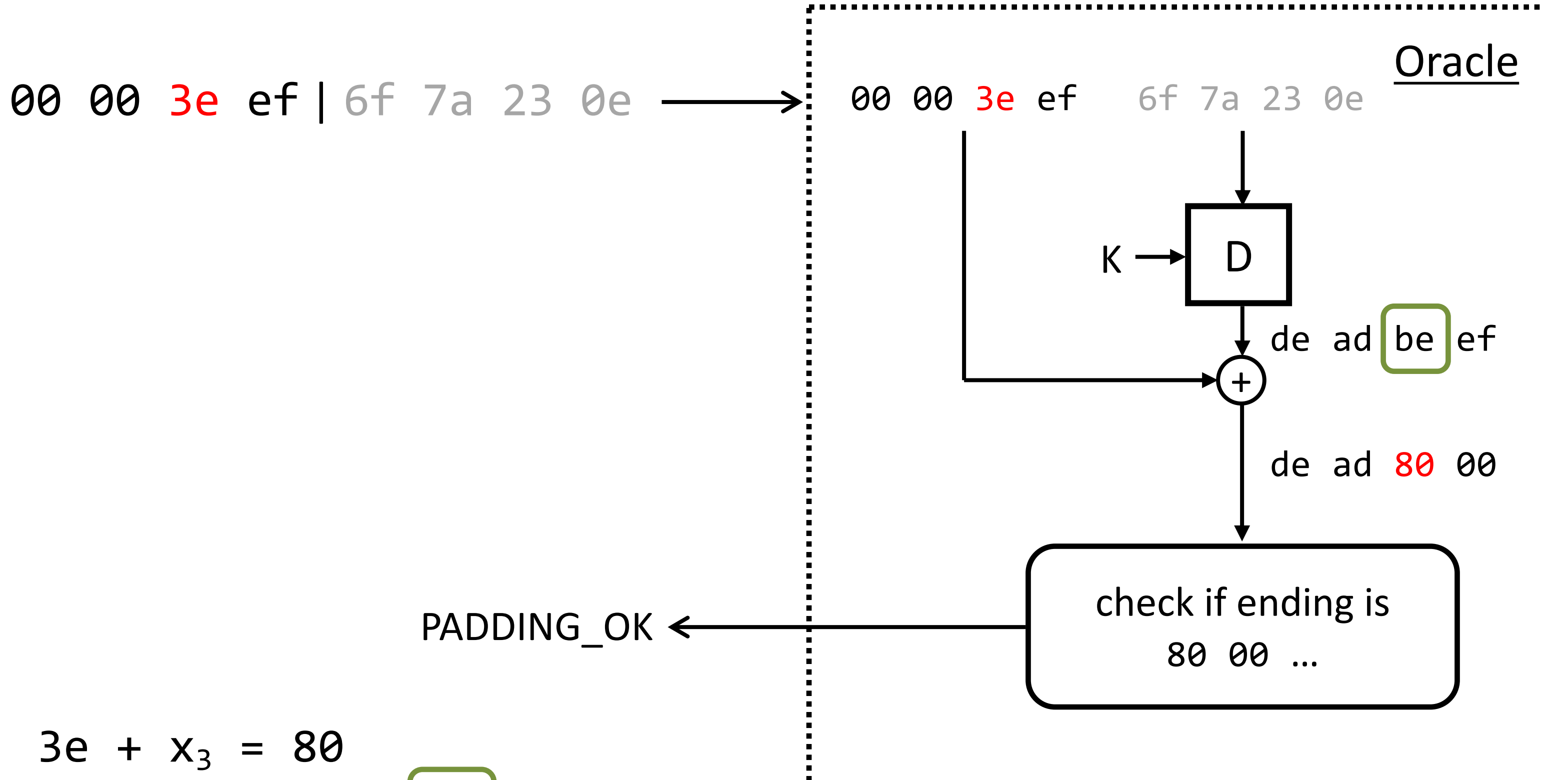
Example

X = de ad be ef
Y = 6f 7a 23 0e
R = 00 00 01 ef



Example

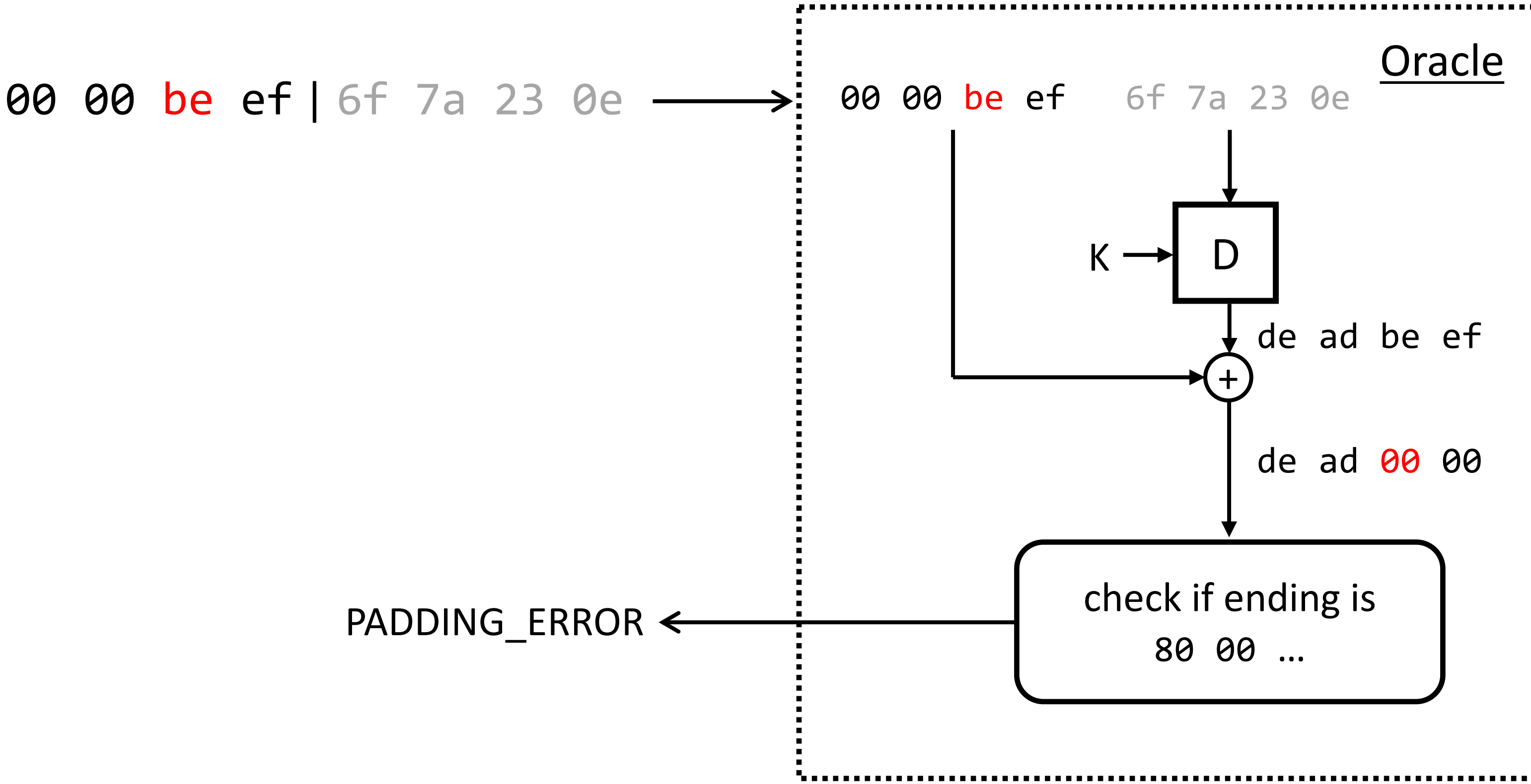
X = de ad be ef
Y = 6f 7a 23 0e
R = 00 00 3e ef



$3e + x_3 = 80$
$x_3 = 3e + 80 = \boxed{be}$

Example

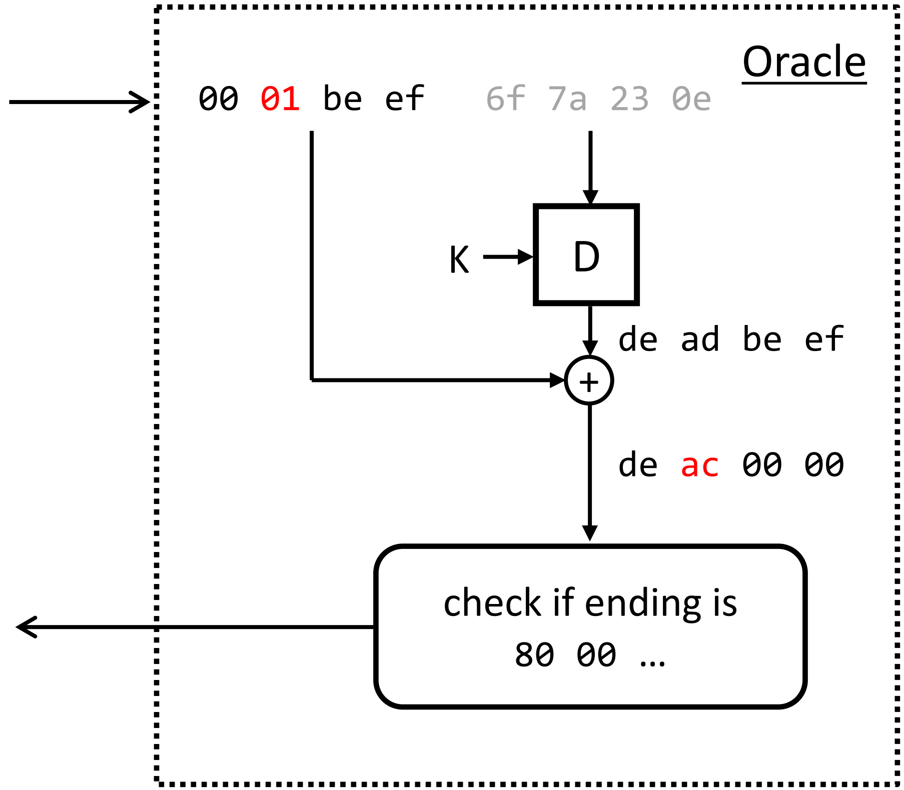
```
X = de ad be ef
Y = 6f 7a 23 0e
R = 00 00 be ef
```



Example

```
X = de ad be ef
Y = 6f 7a 23 0e
R = 00 01 be ef
```

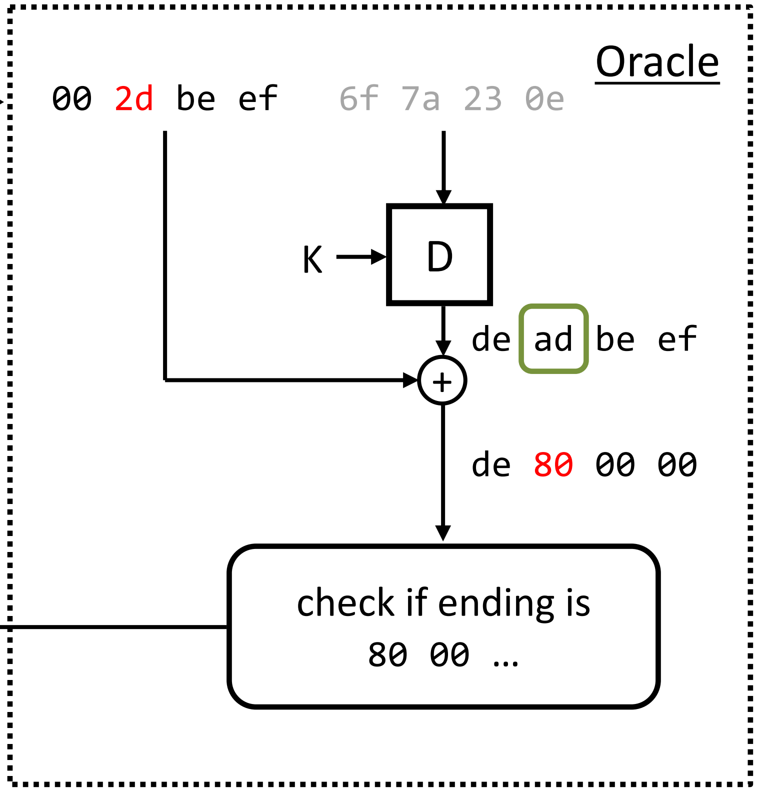
00 01 be ef | 6f 7a 23 0e



Example

```
X = de ad be ef
Y = 6f 7a 23 0e
R = 00 2d be ef
```

00 2d be ef | 6f 7a 23 0e



PADDING_OK

$2d + x_2 = 80$

$x_2 = 2d + 80 = \boxed{ad}$

Attack complexity

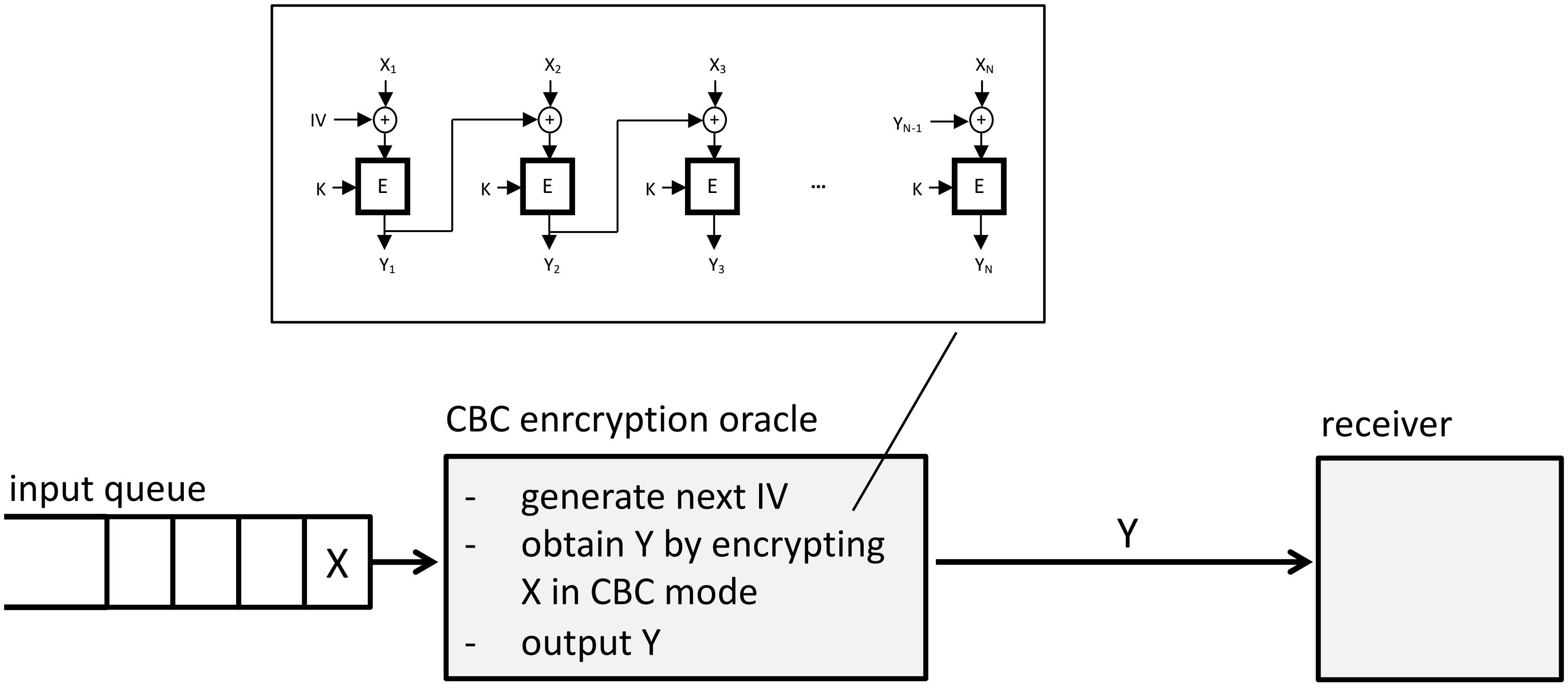
- let's assume the block length of E is b bytes
- we measure processing complexity in the number of calls to the Oracle
- computing the last byte(s) requires
  - at most 256+b calls
  - on average 128+b calls
- the most likely case is that the number of remaining bytes is b-1
- computing each remaining byte requires
  - at most 256 calls
  - on average 128 calls
- so the complexity is
  - worst case: 256 + b + (b-1)*256 = b*257
  - average: 128 + b + (b-1)*128 = b*129
- e.g., in case of AES, b = 16:
  - worst case complexity: 4112 calls
  - Average complexity: appr. 2064 calls

Exploiting Predictable IVs

The problem of predictable IVs

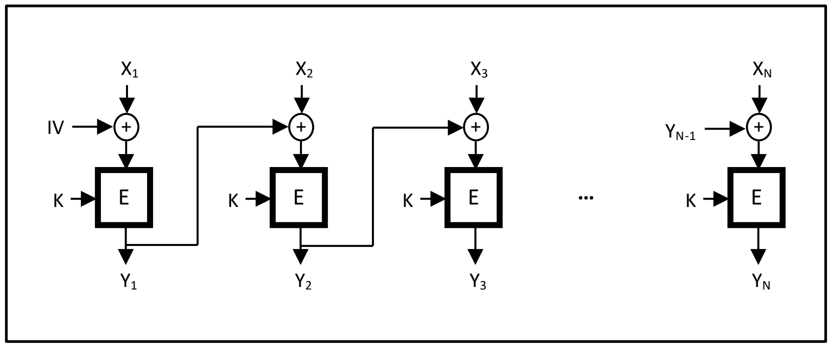
- let $Y_i = E_K(Y_{i-1} + X_i)$ for some i (part of a CBC encrypted message)
- we want to obtain X_i
- let us assume that we have access to a CBC encryption oracle (**chosen plaintext attack model**) and the oracle uses predictable IVs
- so let's predict the next IV, and submit a plaintext with $IV + Y_{i-1} + X^*$ as the first block to the oracle, where X^* is our guess for X_i
- the oracle outputs a ciphertext with $E_K(IV + IV + Y_{i-1} + X^*) = E_K(Y_{i-1} + X^*)$ as the first block
- if our guess was correct (i.e., $X_i = X^*$), then the above first block is equal to Y_i
- if not, we can try agian with another guess, until we'll have the right one

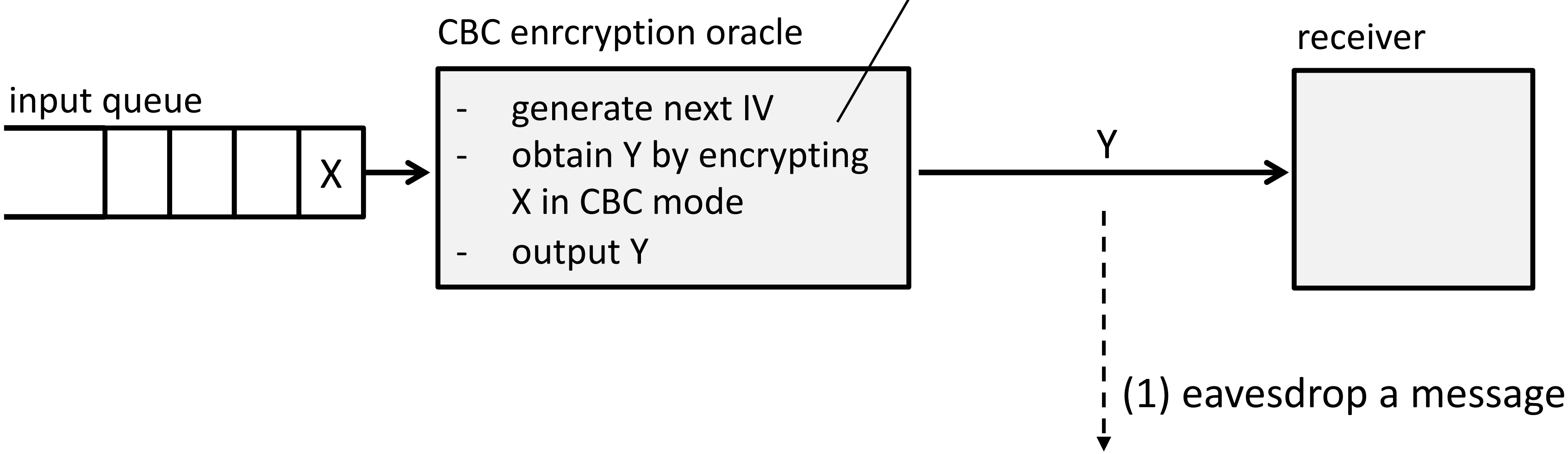
Exploiting predictable IVs



- chosen plaintext assumptions:
  – the attacker can inject messages into the input queue (choose X)
  – the attacker can eavesdrop the communication channel (obtain Y)
- predictable IV assumption:
  – the attacker can predict the value of the next IV to be used by the oracle

Exploiting predictable IVs

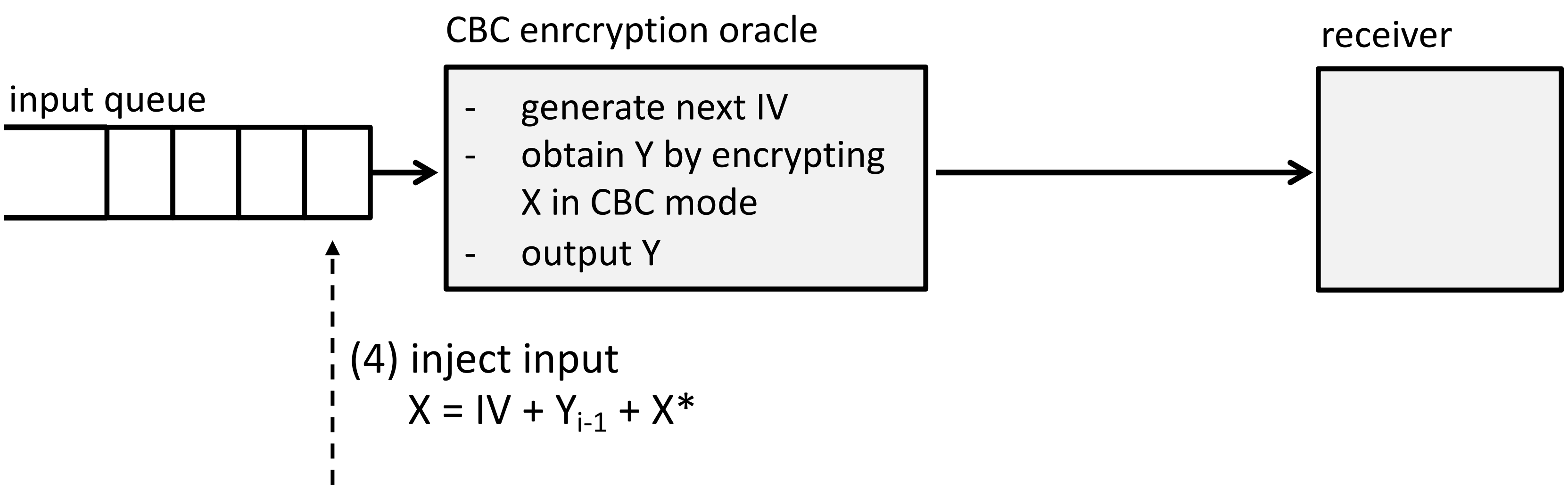




$Y = Y_1\ Y_2\ \ldots\ Y_N$

$Y_i = E_K(Y_{i-1} + X_i)$

Exploiting predictable IVs



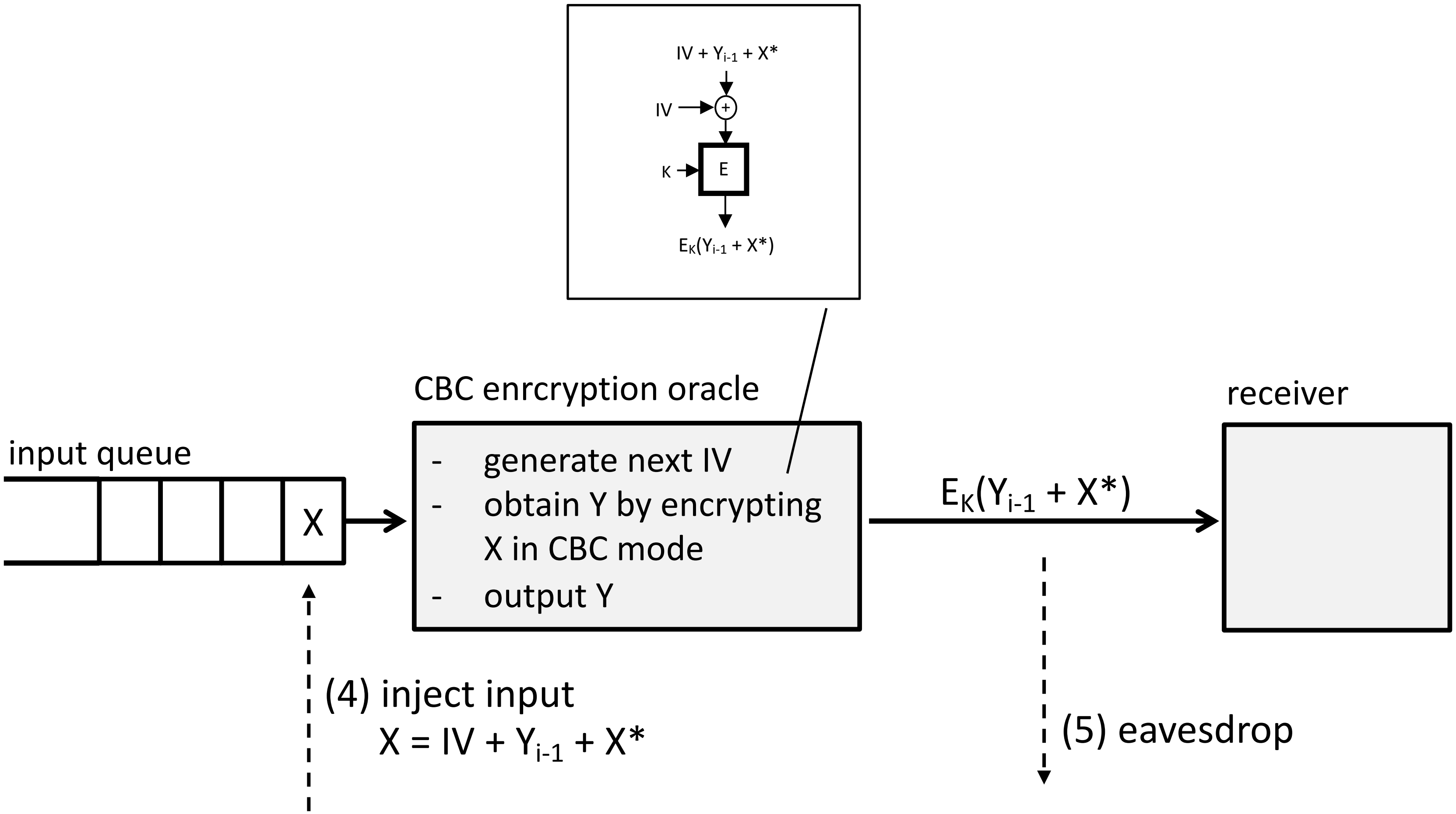
(2) predict next IV: IV

(3) make a guess X^* for X_i

$Y = Y_1\, Y_2 \ldots Y_N$

$Y_i = E_K(Y_{i-1} + X_i)$

Exploiting predictable IVs



(6) compare $E_K(Y_{i-1} + X^*)$ to $E_K(Y_{i-1} + X_i)$

if equal, then X_i must be X*

otherwise repeat from step (2)

$Y = Y_1\, Y_2 \ldots Y_N$

$Y_i = E_K(Y_{i-1} + X_i)$

Exploiting predictable IVs in practice

- in practice, the block length of the cipher is large and guessing the value of X_i is infeasible
- what if we don't need to guess the entire block, because large part of it is already known?
- then predictable IVs can still be a problem!

Lessons learned

- content leak problem
  → use a sufficiently large block size (e.g., 128 bits) or encrypt sufficiently small chunks of data with the same key
- padding oracle attack
  → avoid leaking information about the correctness of the padding
    → explicit error messages should be avoided
    → pay attention to side channels as well (e.g., timing of oracle response)
- exploiting predictable IVs
  → don't use predictable IVs; there are methods to generate IVs that are unpredictable for an attacker

Control questions

- What is the basic idea behind the content leak problem?
- When do we expect to have at least two identical ciphertext blocks in a CBC encrypted message? (length of message as a function of the block length)
- What attacker model does the padding oracle attack belong to?
- What is the main idea of the padding oracle attack?
- How we can prevent padding oracle attacks?
- Why are predictable IVs in CBC mode dangerous?
- What could be the problem with repeated guessing of a plaintext block in practice?
- When can the guessing attack that exploits predictable IVs still work?