

# Password Interception in a SSL/TLS Channel

Brice Canvel (NagraCard)

Alain Hiltgen (UBS)

Serge Vaudenay (EPFL)

Martin Vuagnoux (EPFL and Ilion)

<http://lasecwww.epfl.ch/>



ÉCOLE POLYTECHNIQUE  
FÉDÉRALE DE LAUSANNE

# Contents

---

★ TLS and CBC-PAD

# Contents

---

- ★ TLS and CBC-PAD
- ★ Side Channel Attack against CBC-PAD

# Contents

---

- ★ TLS and CBC-PAD
- ★ Side Channel Attack against CBC-PAD
- ★ Timing Attack

# Contents

---

- ★ TLS and CBC-PAD
- ★ Side Channel Attack against CBC-PAD
- ★ Timing Attack
- ★ Multisession Attack

# Contents

---

- ★ TLS and CBC-PAD
- ★ Side Channel Attack against CBC-PAD
- ★ Timing Attack
- ★ Multisession Attack
- ★ Experiments and Discussions

# TLS and CBC-PAD

# A Typical TLS Session

---

Client

Server

# A Typical TLS Session

---

Client

Server

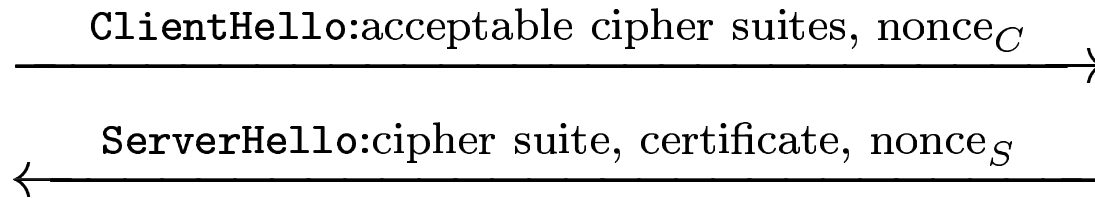
ClientHello:acceptable cipher suites, nonce<sub>C</sub>



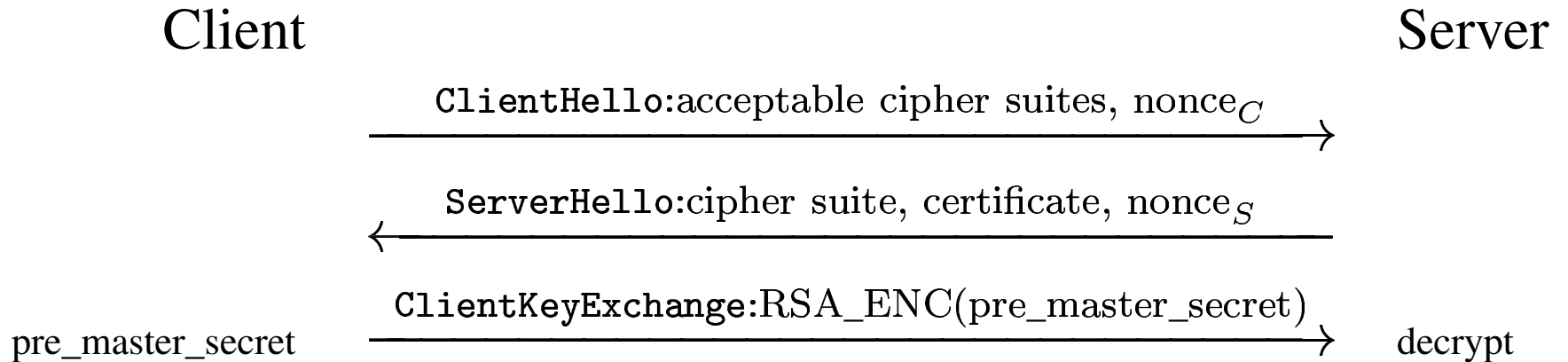
# A Typical TLS Session

Client

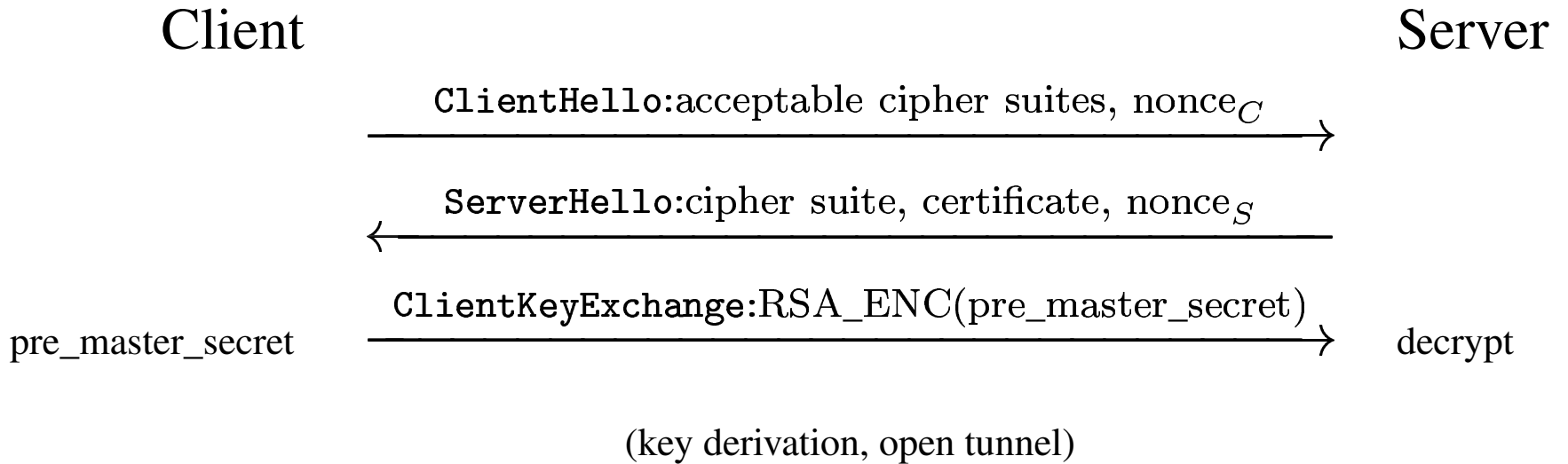
Server



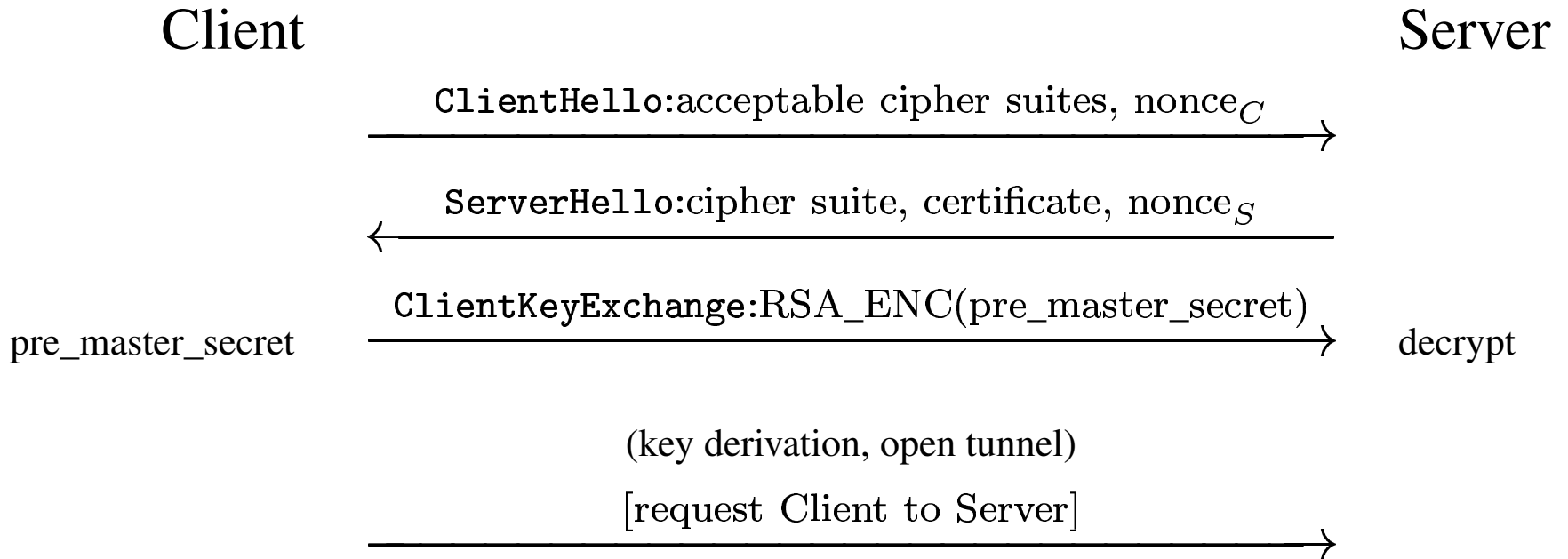
# A Typical TLS Session



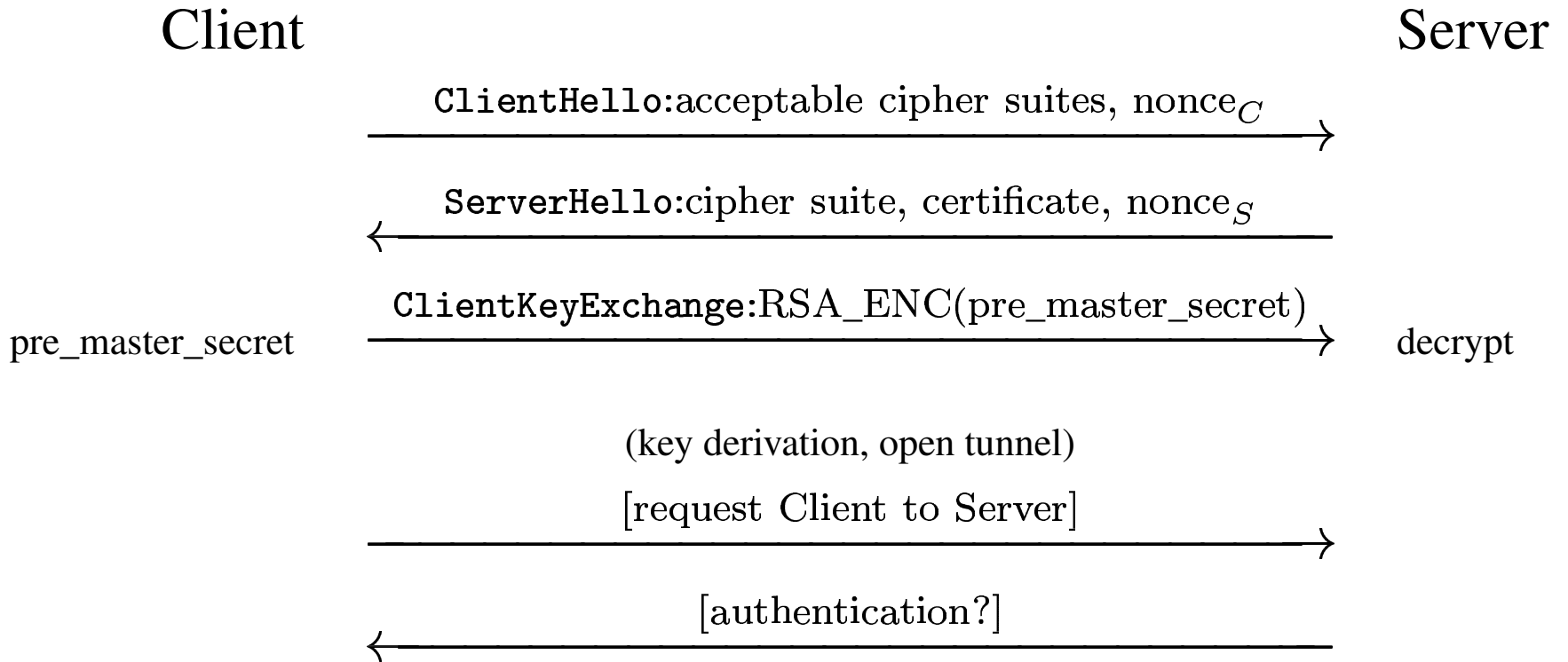
# A Typical TLS Session



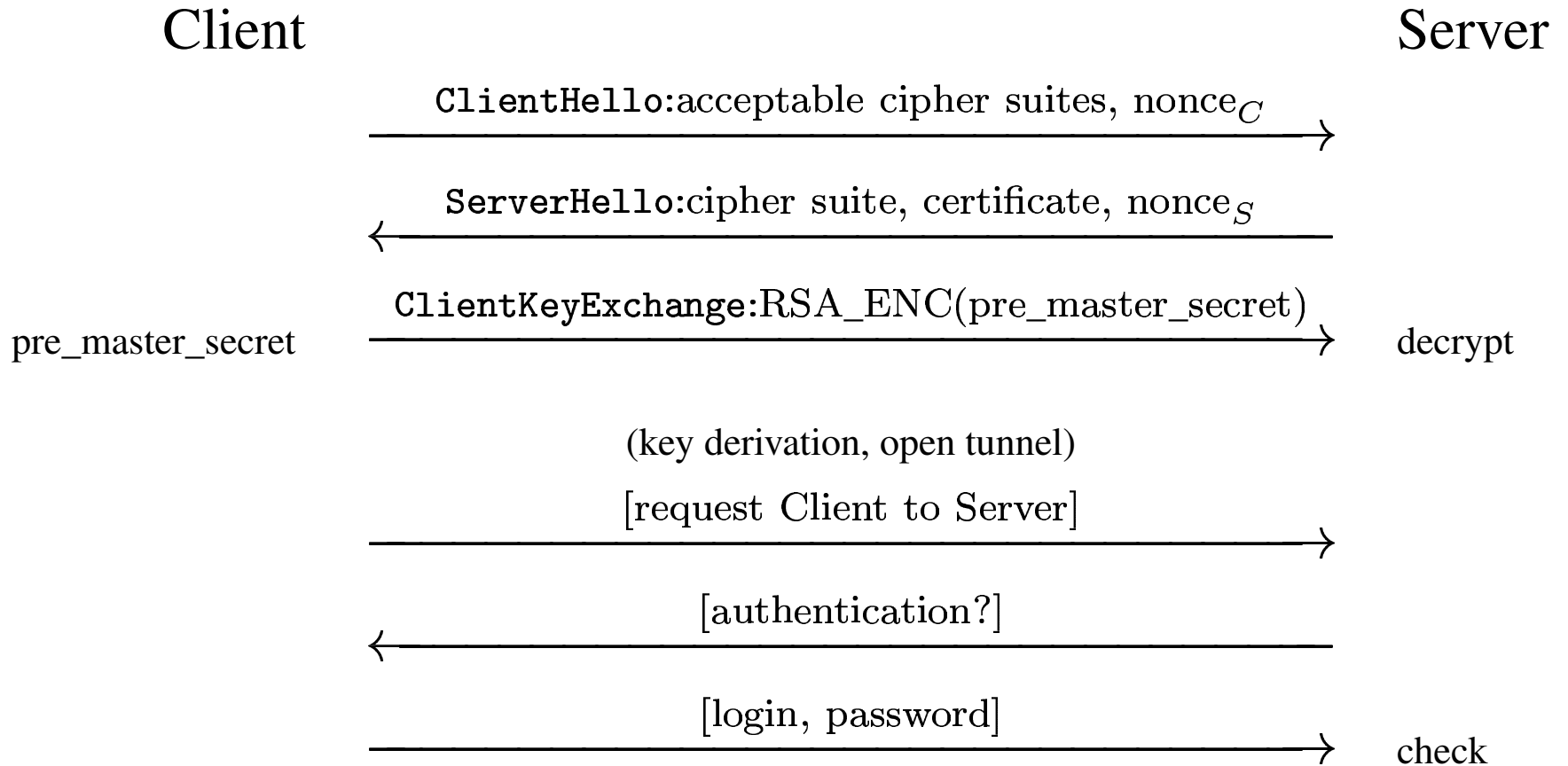
# A Typical TLS Session



# A Typical TLS Session

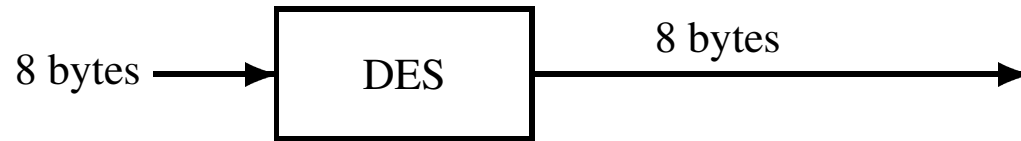


# A Typical TLS Session

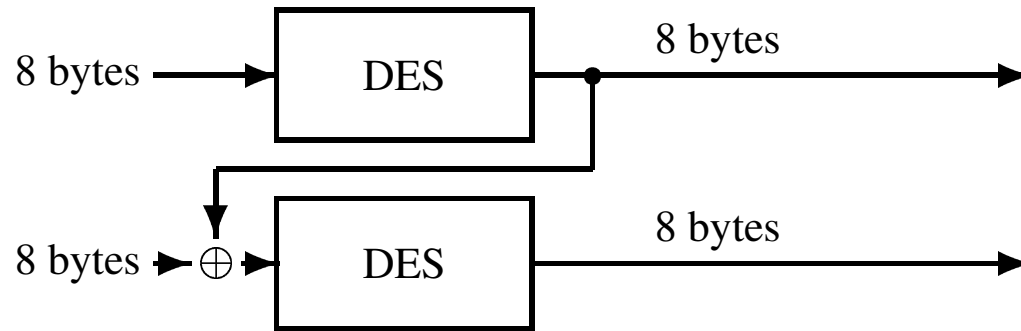


# CBC-PAD

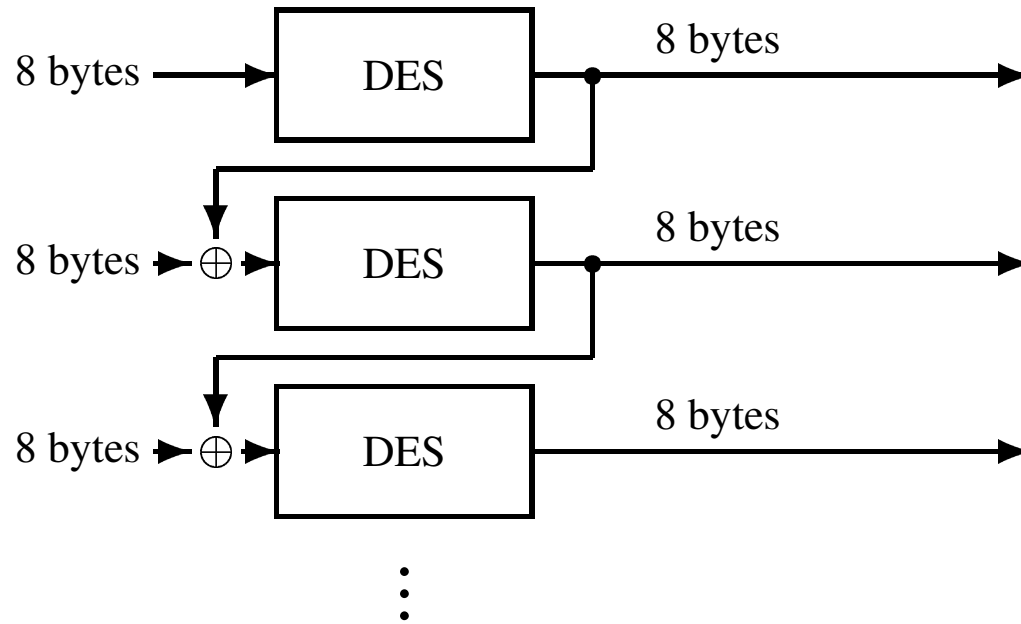
---



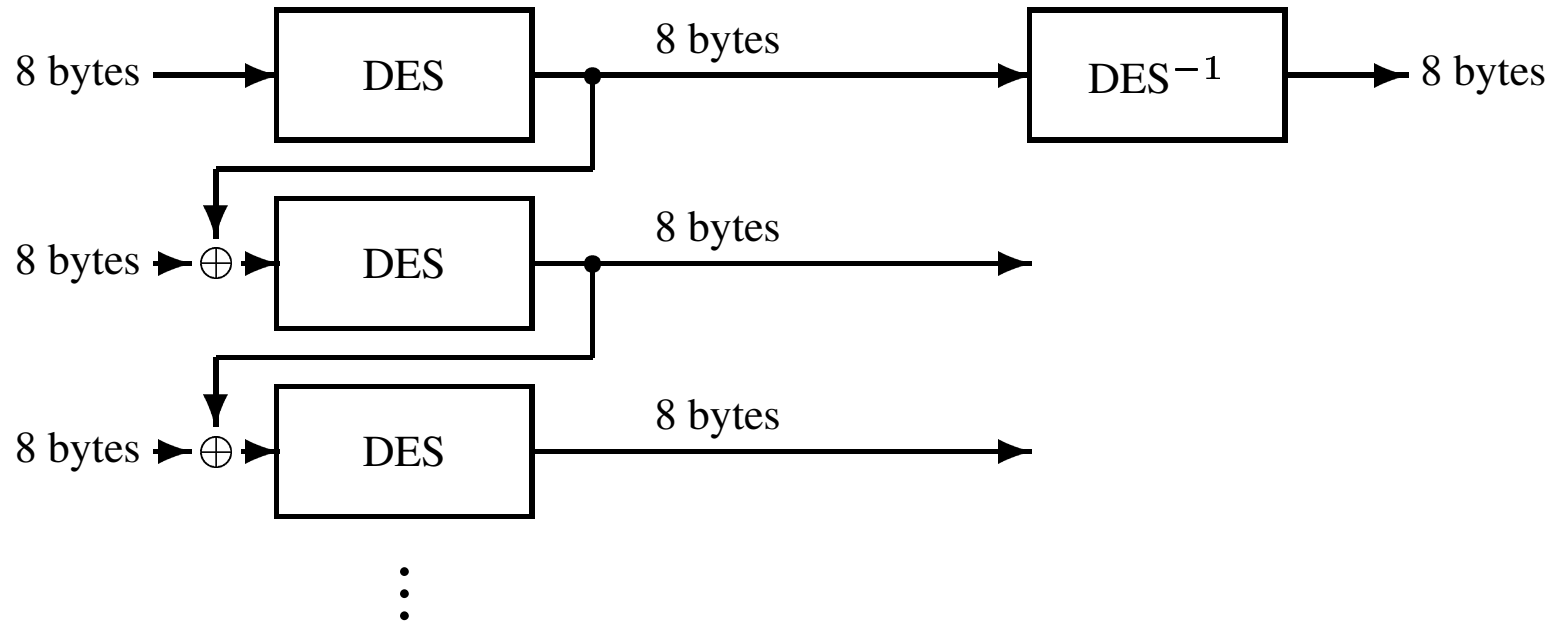
# CBC-PAD



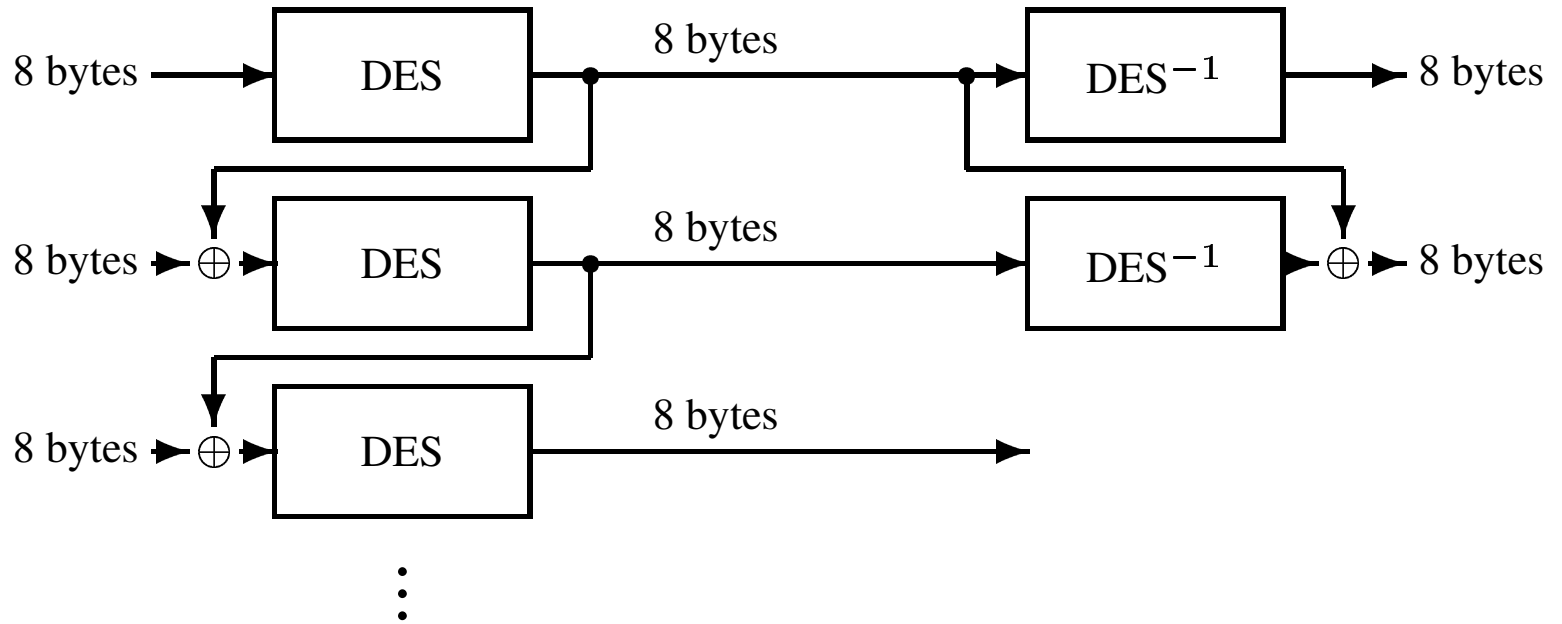
# CBC-PAD



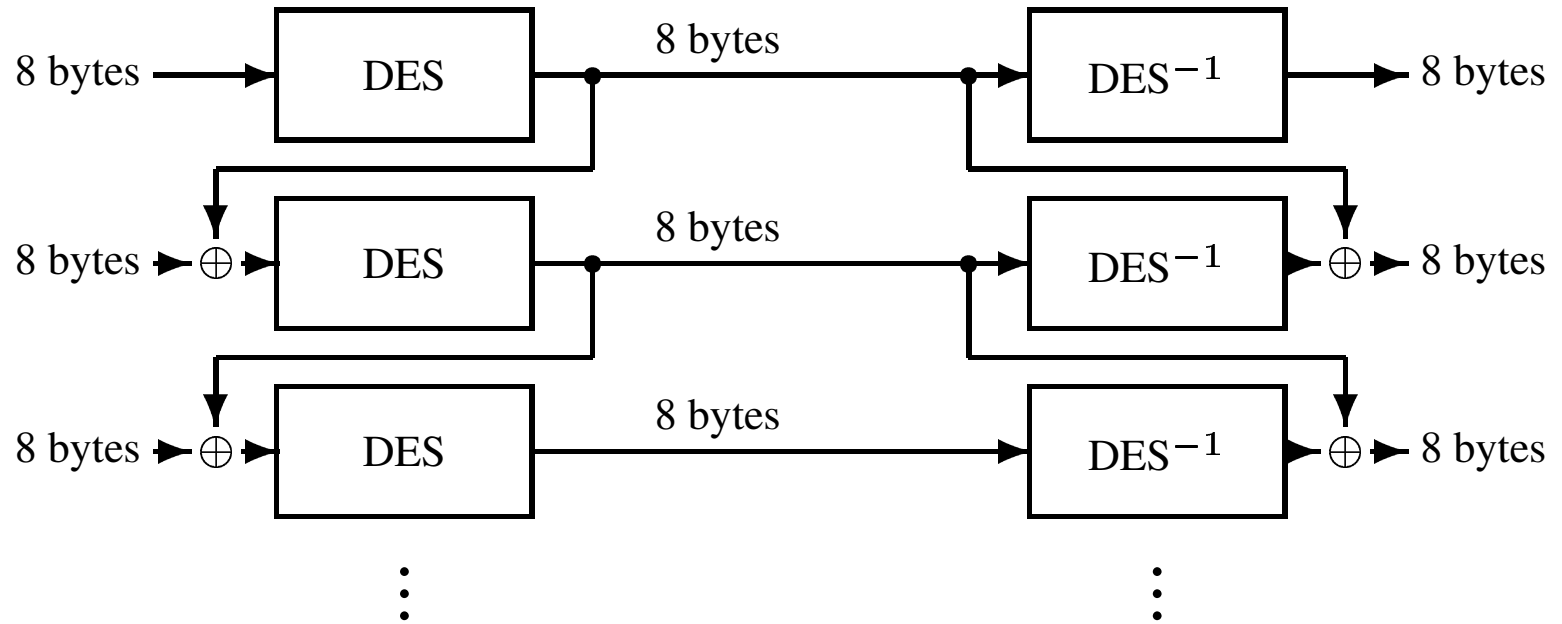
# CBC-PAD



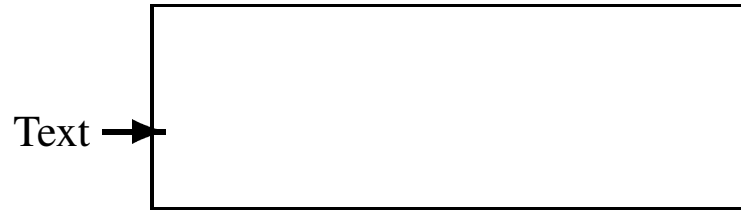
# CBC-PAD



# CBC-PAD

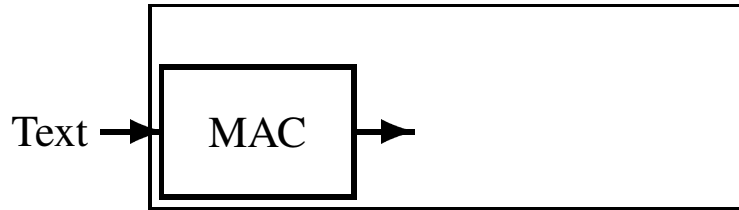


# TLS and CBC-PAD : Example



C R Y P T O ' O  
3 U C S B

# TLS and CBC-PAD : Example

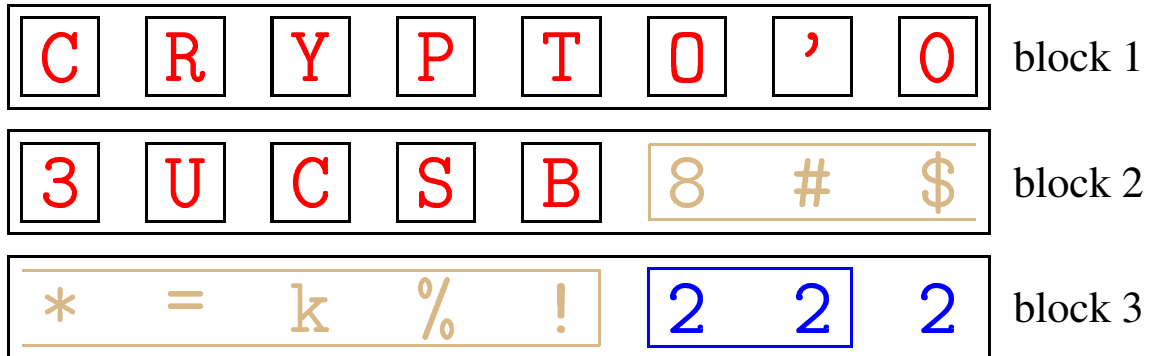
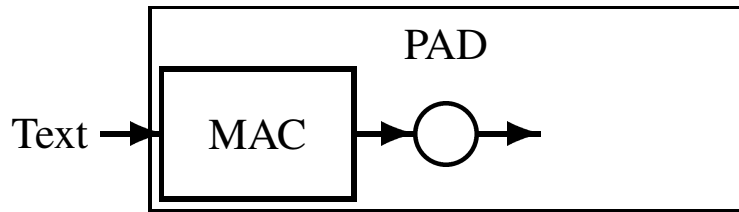


C R Y P T O ' O block 1

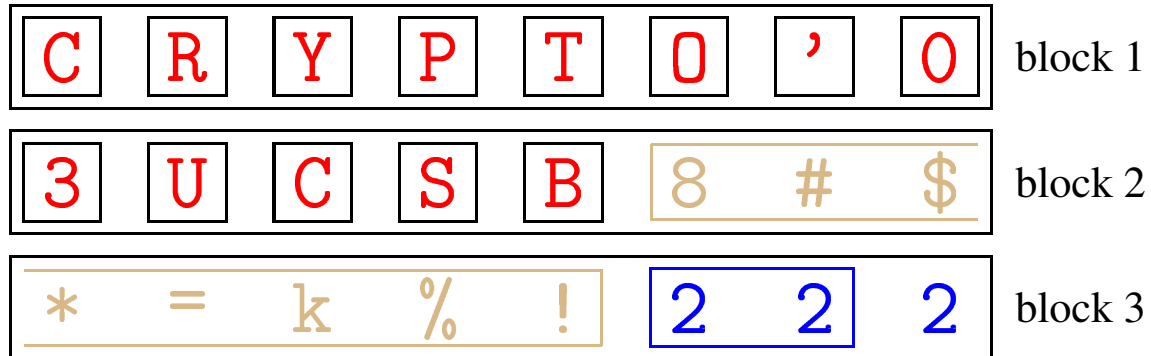
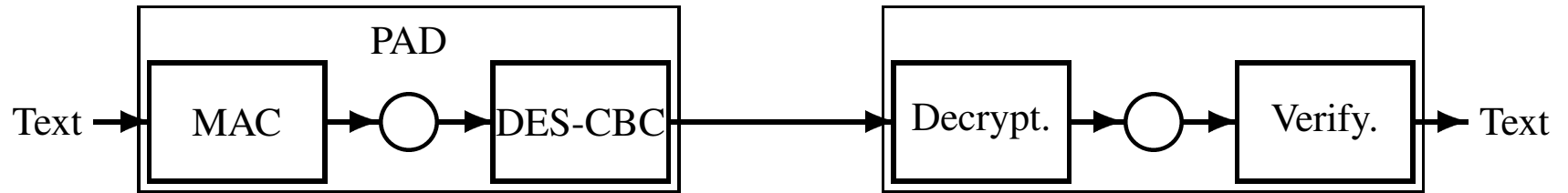
3 U C S B 8 # \$ block 2

\* = k % !

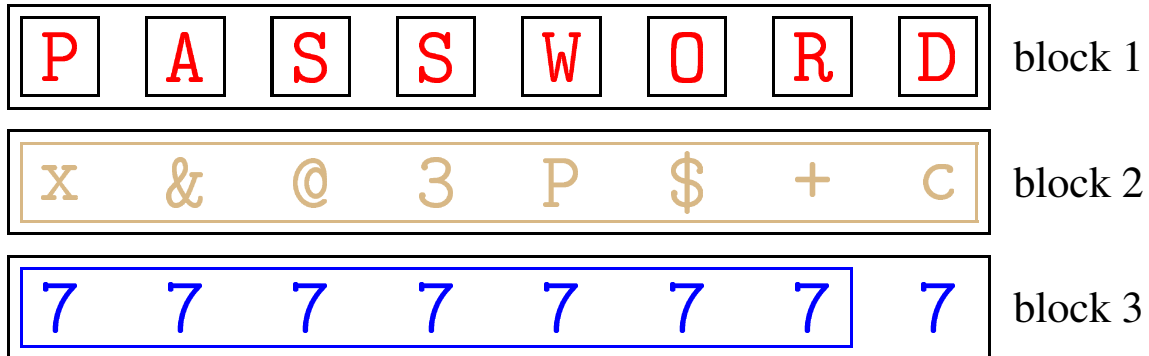
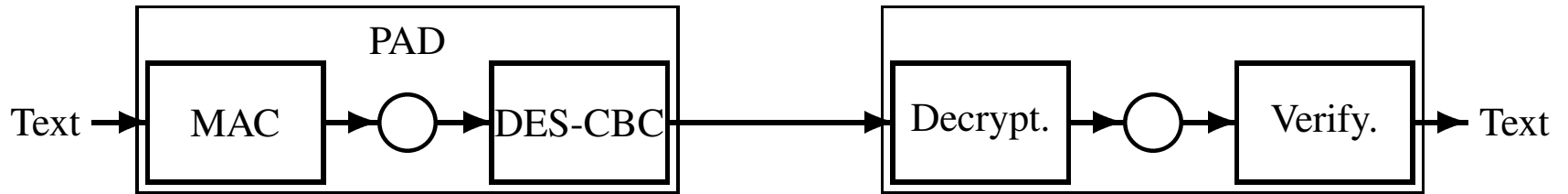
# TLS and CBC-PAD : Example



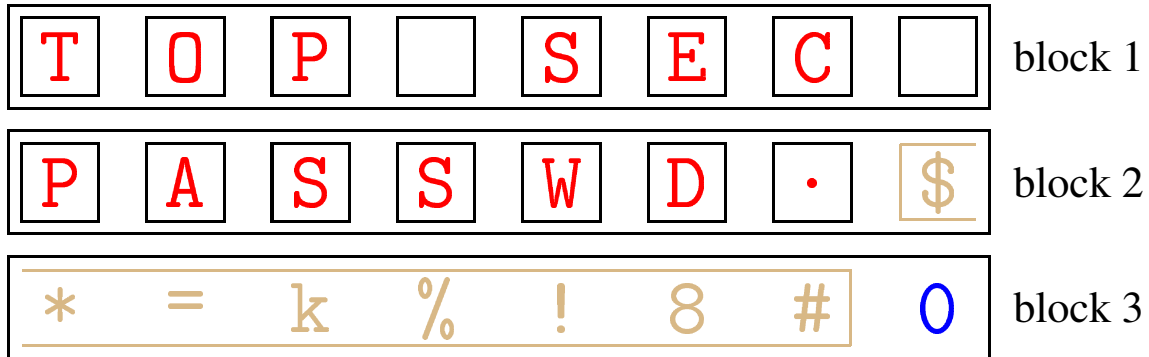
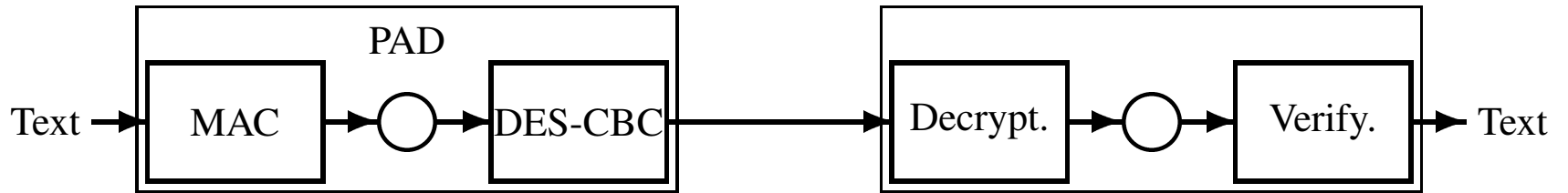
# TLS and CBC-PAD : Example



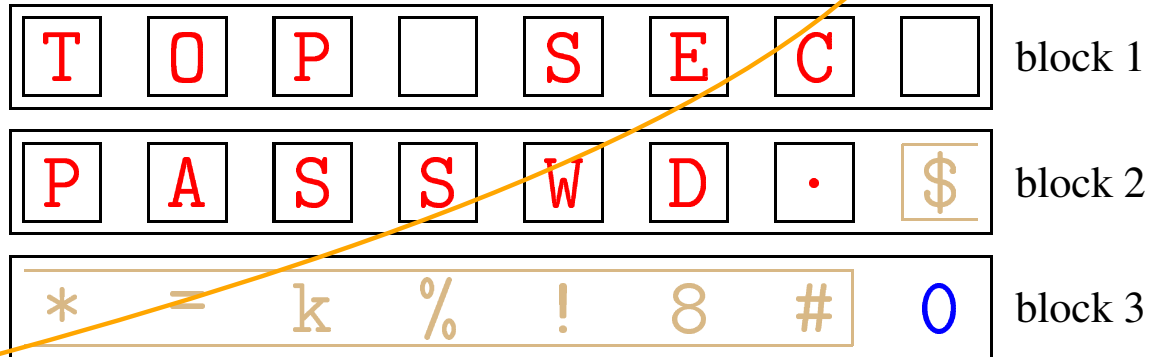
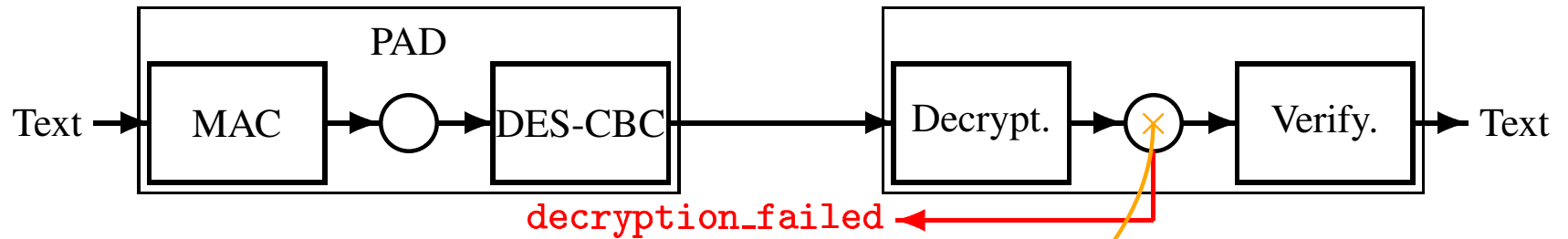
# TLS and CBC-PAD : Example



# TLS and CBC-PAD : Example

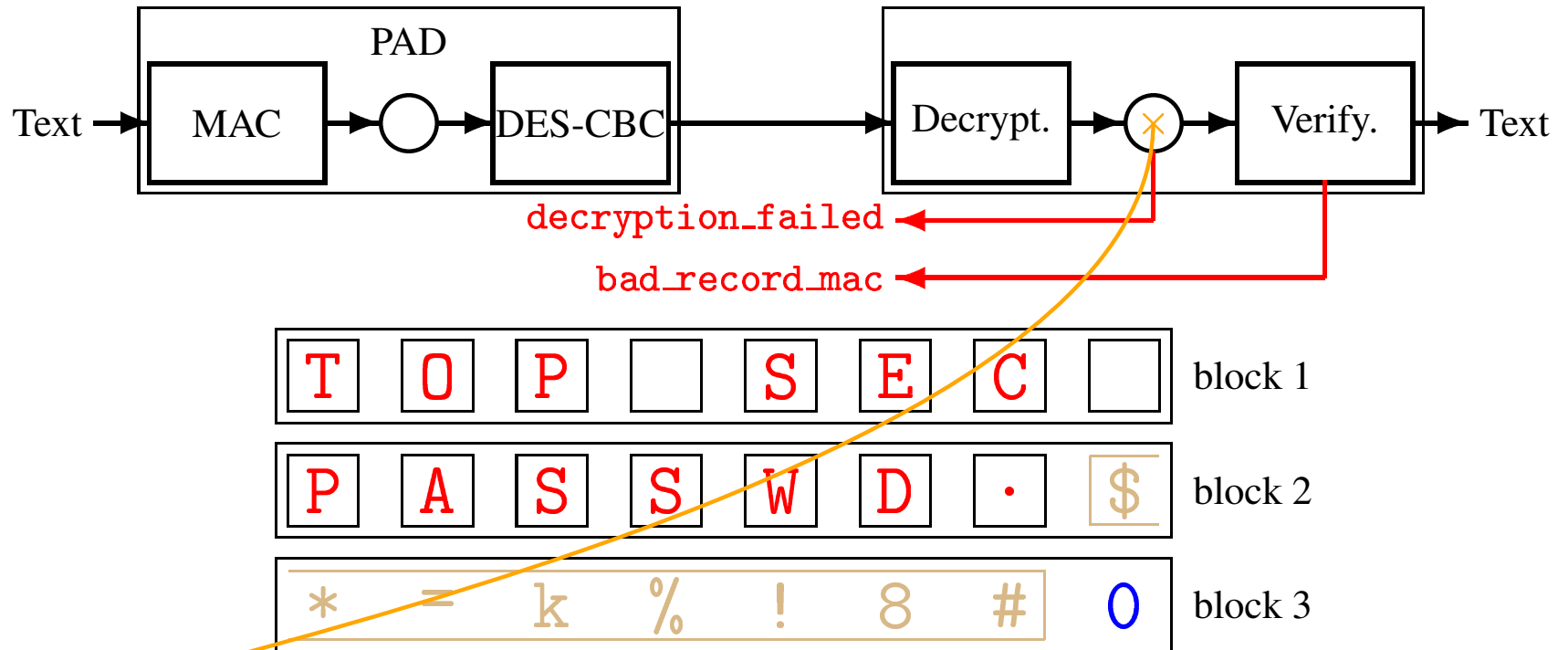


# TLS and CBC-PAD : Example



if the last byte equals  $n$ , verify that  $n + 1$  bytes equal  $n$

# TLS and CBC-PAD : Example



if the last byte equals  $n$ , verify that  $n + 1$  bytes equal  $n$

# TLS and CBC-PAD : Notes

---

- ★ In TLS, error messages `decryption_failed` and `bad_mac_error` are fatal and abort the session

# TLS and CBC-PAD : Notes

---

- ★ In TLS, error messages `decryption_failed` and `bad_mac_error` are fatal and abort the session
- ★ Error messages in TLS are also MACed, padded and then encrypted before being sent

# Side Channel Attack against CBC-PAD

## Side Channel Attack against CBC-PAD

---

- ★ Attack found by Serge Vaudenay and presented at Eurocrypt'02

## Side Channel Attack against CBC-PAD

---

- ★ Attack found by Serge Vaudenay and presented at Eurocrypt'02
- ★ Enables decryption of blocks

## Side Channel Attack against CBC-PAD

---

- ★ Attack found by Serge Vaudenay and presented at Eurocrypt'02
- ★ Enables decryption of blocks
- ★ Error messages must be available

## Side Channel Attack against CBC-PAD

---

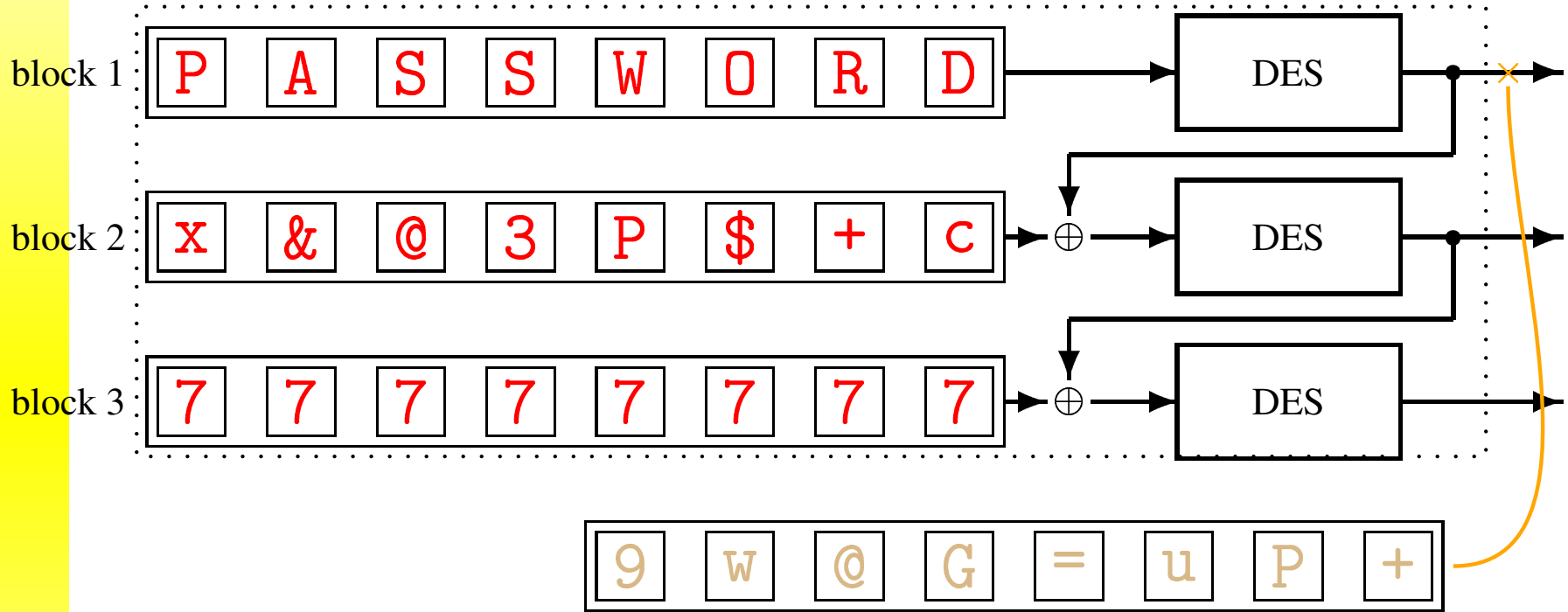
- ★ Attack found by Serge Vaudenay and presented at Eurocrypt'02
- ★ Enables decryption of blocks
- ★ Error messages must be available
- ★ Sessions must not abort

## Side Channel Attack against CBC-PAD

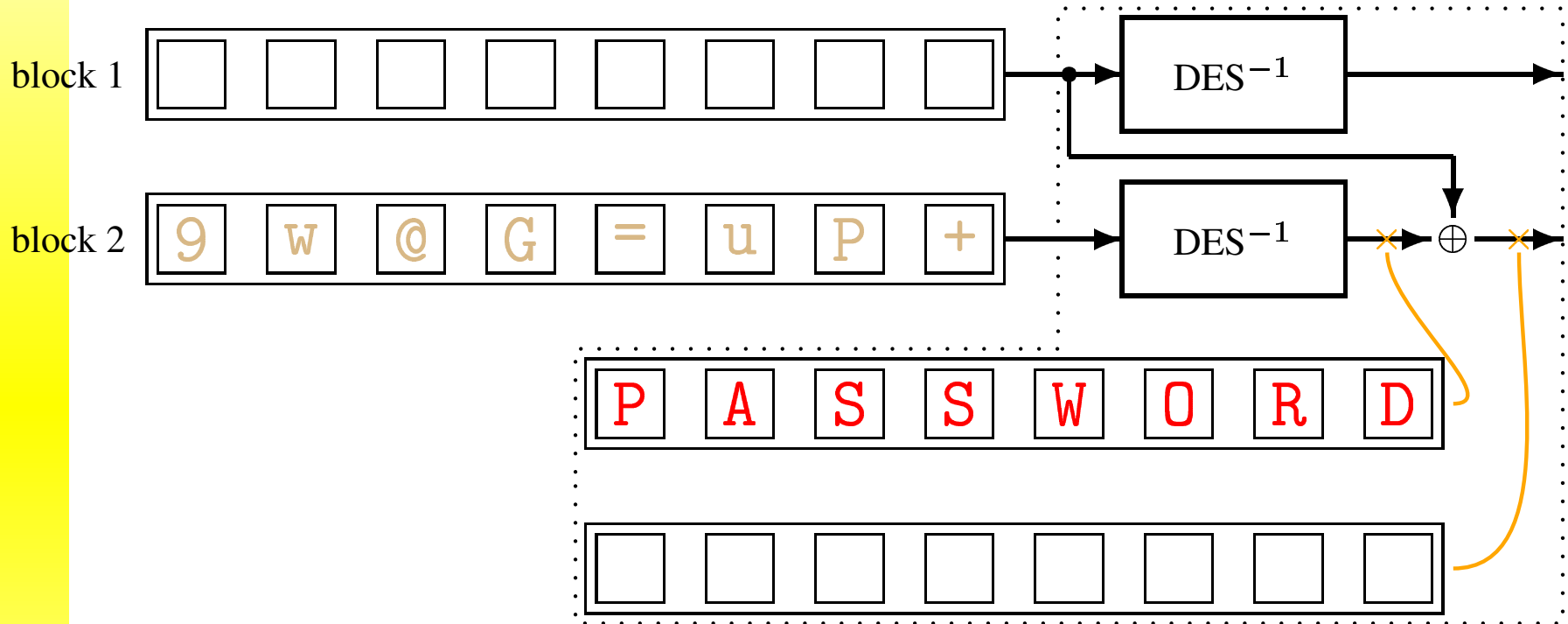
---

- ★ Attack found by Serge Vaudenay and presented at Eurocrypt'02
- ★ Enables decryption of blocks
- ★ Error messages must be available
- ★ Sessions must not abort
- ★ Example : WTLS

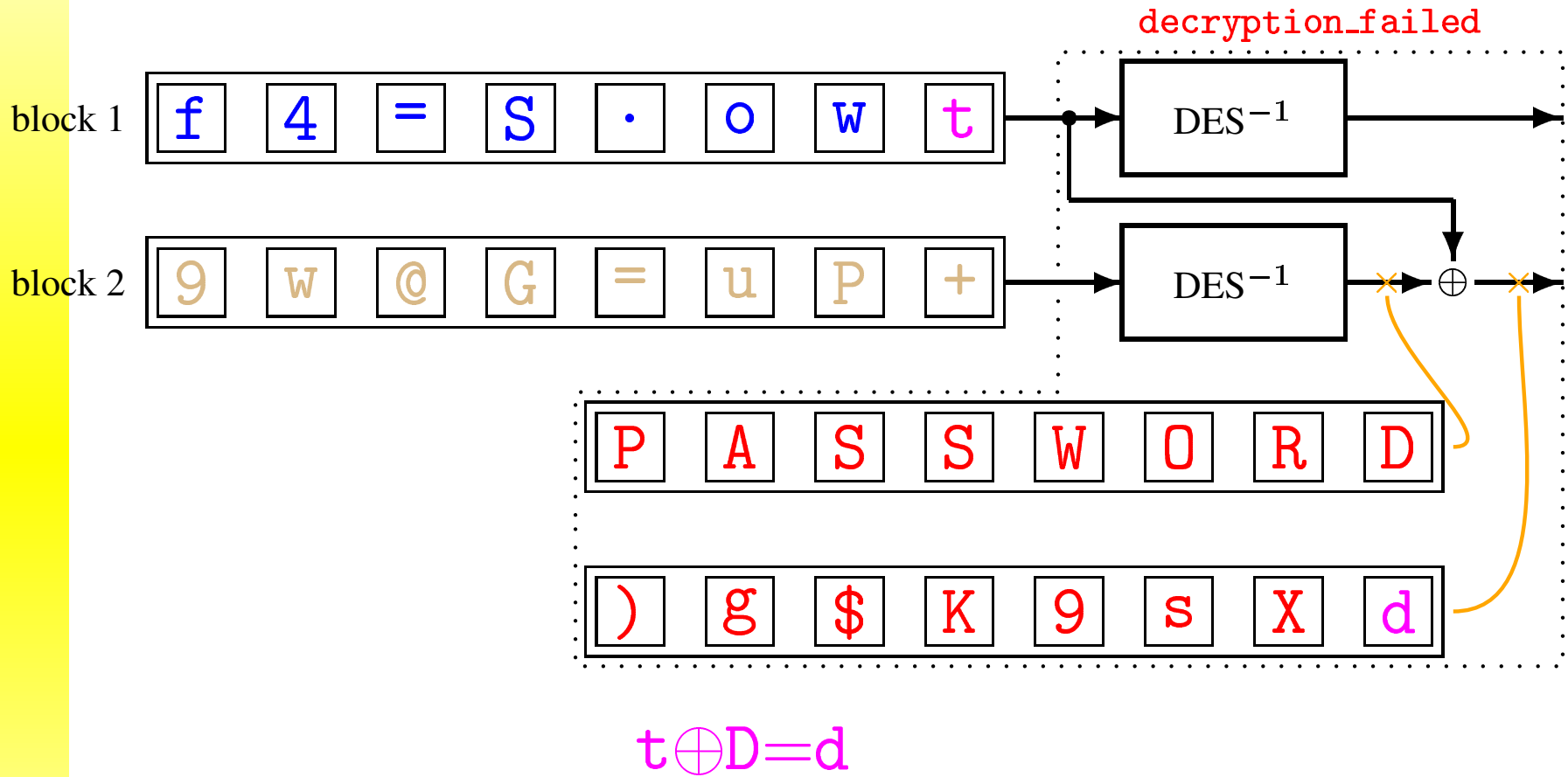
# Side Channel Attack against CBC-PAD



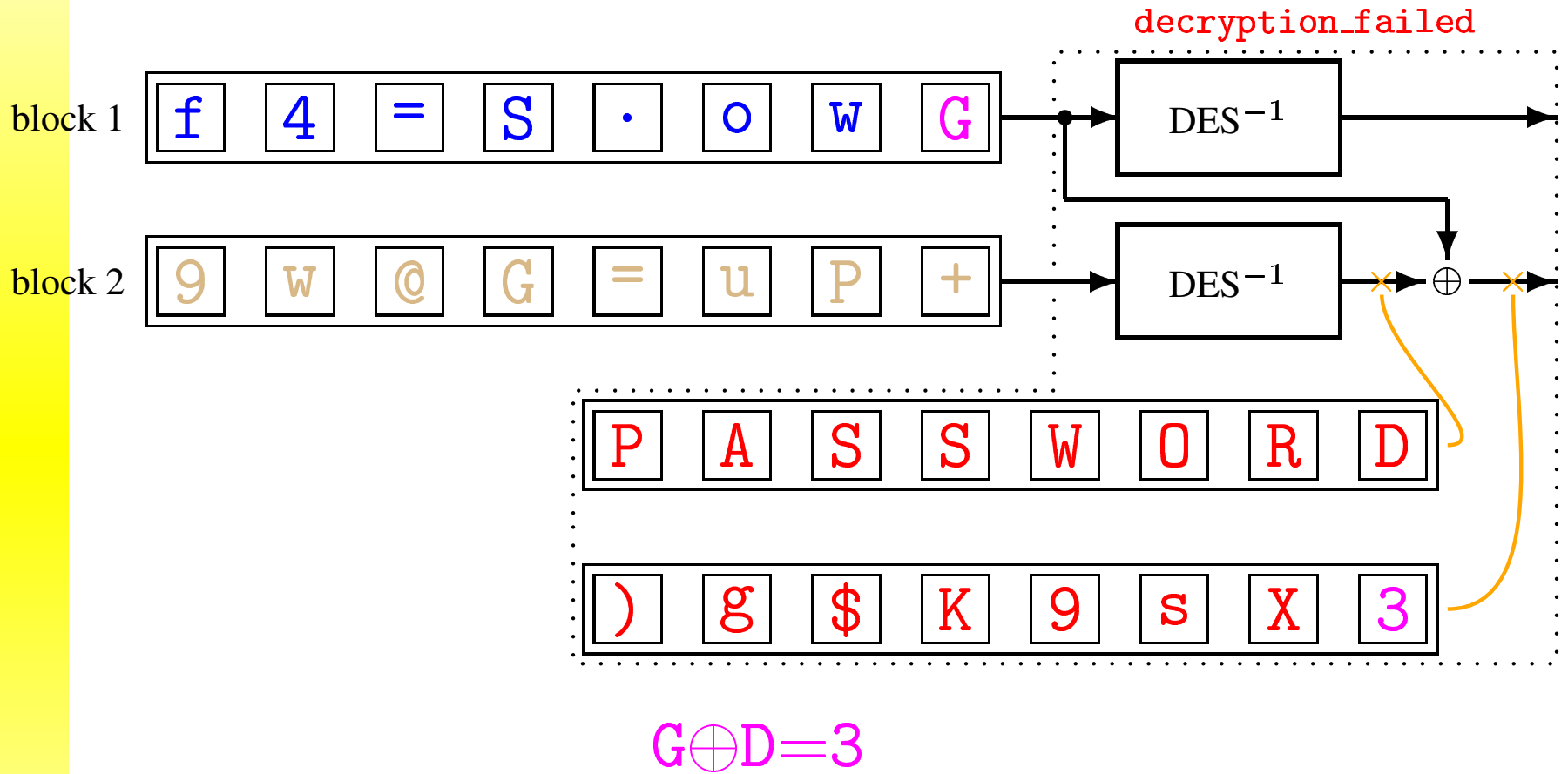
# Side Channel Attack against CBC-PAD



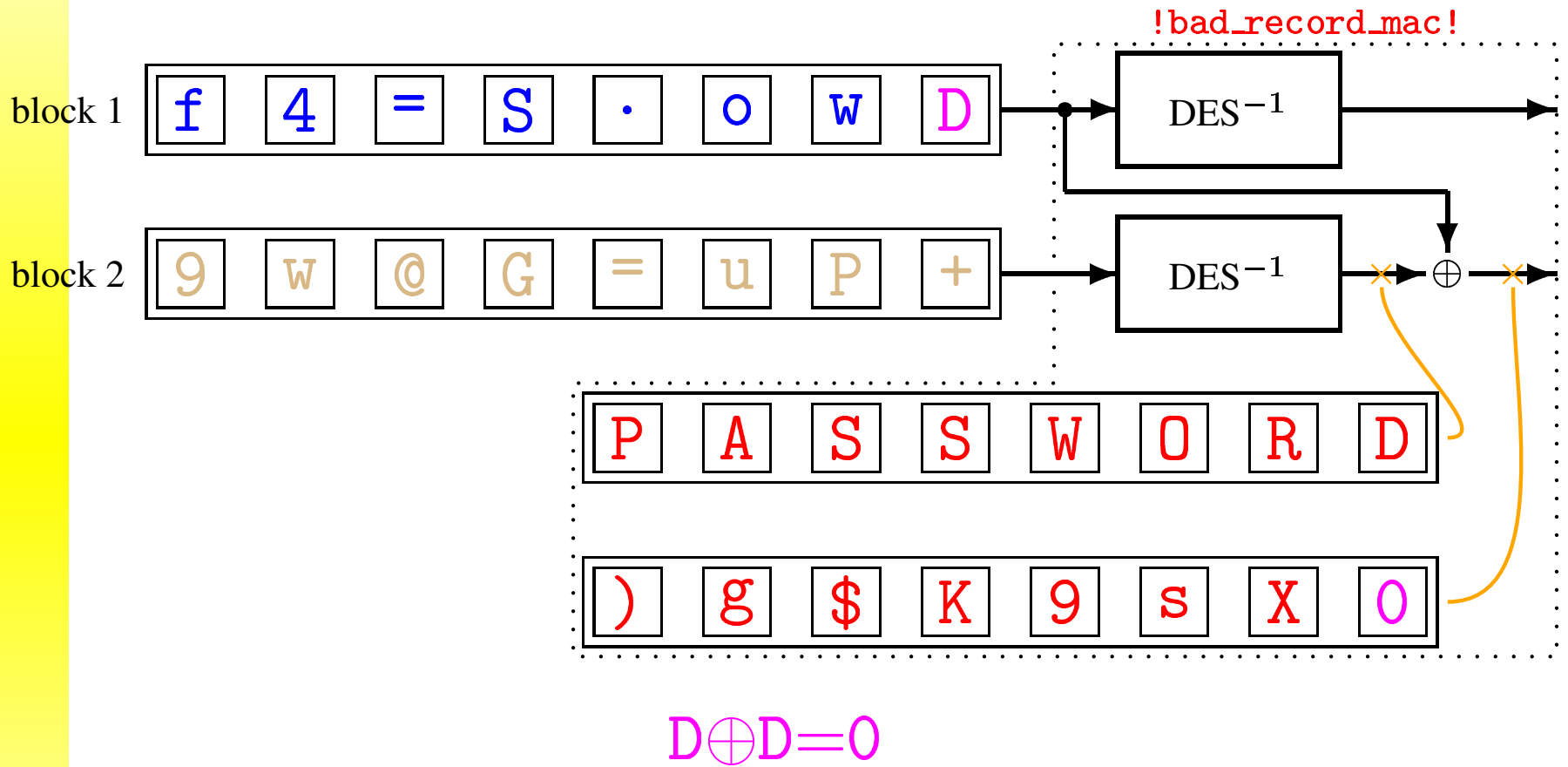
# Side Channel Attack against CBC-PAD



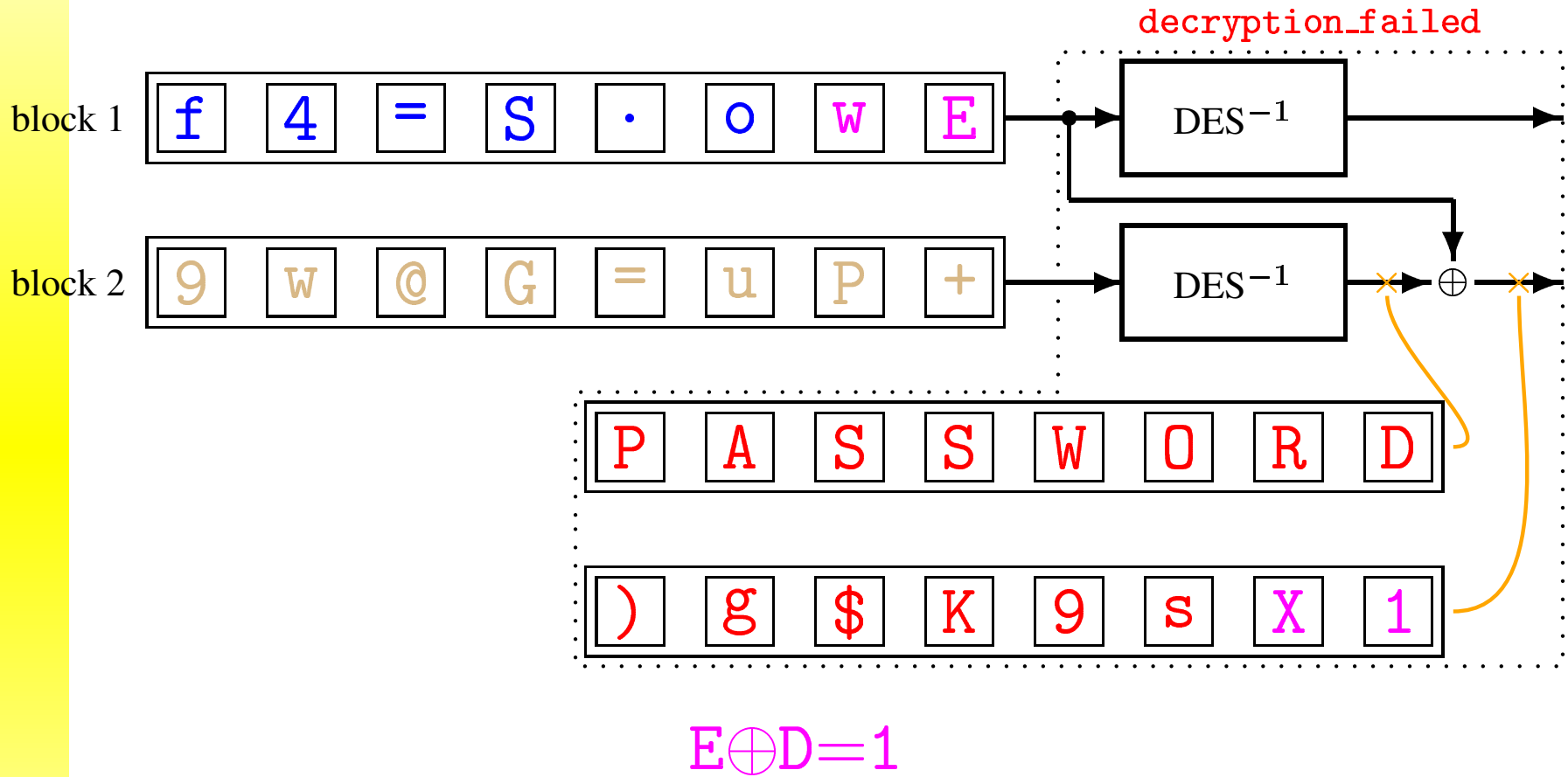
# Side Channel Attack against CBC-PAD



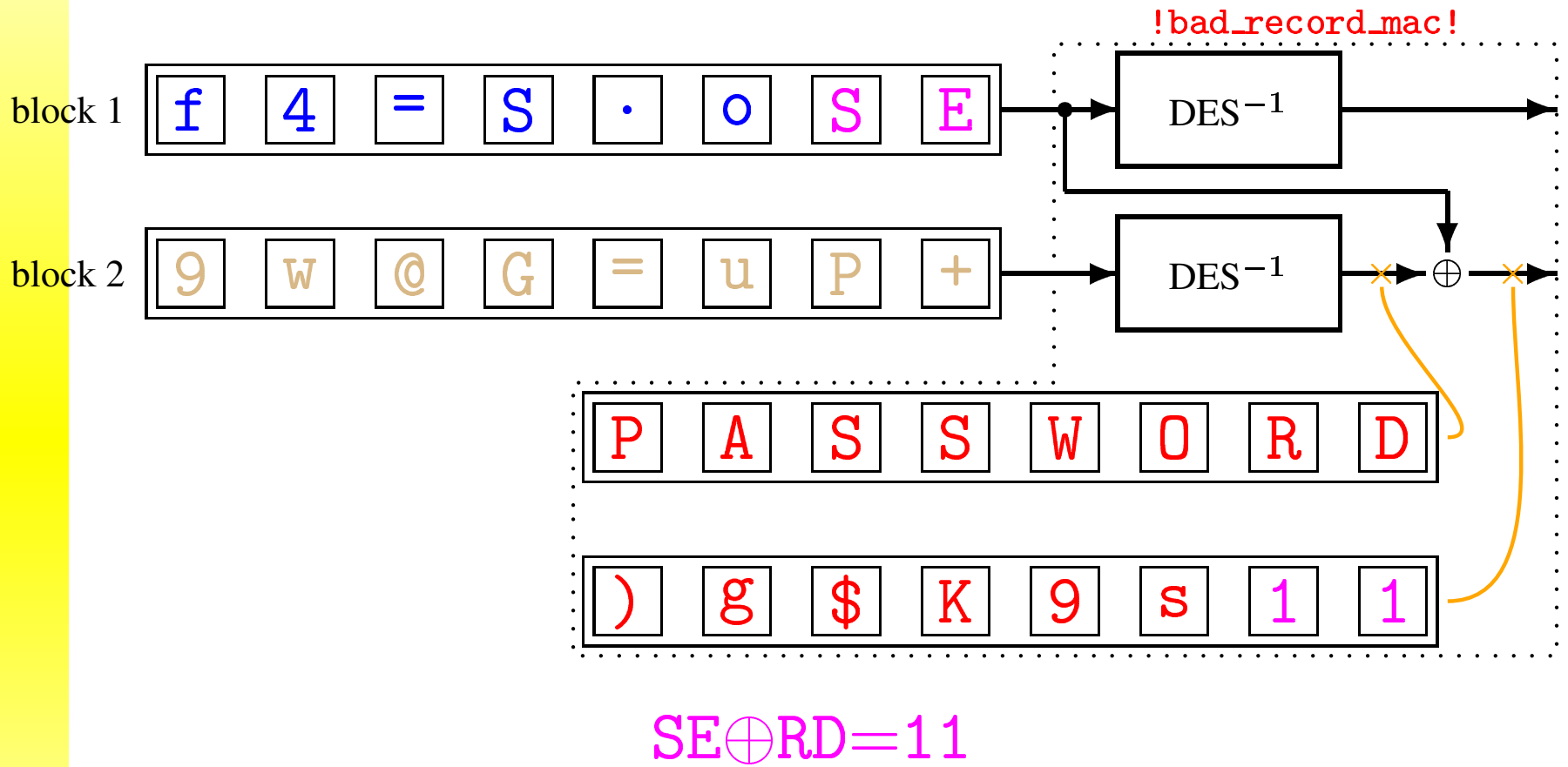
# Side Channel Attack against CBC-PAD



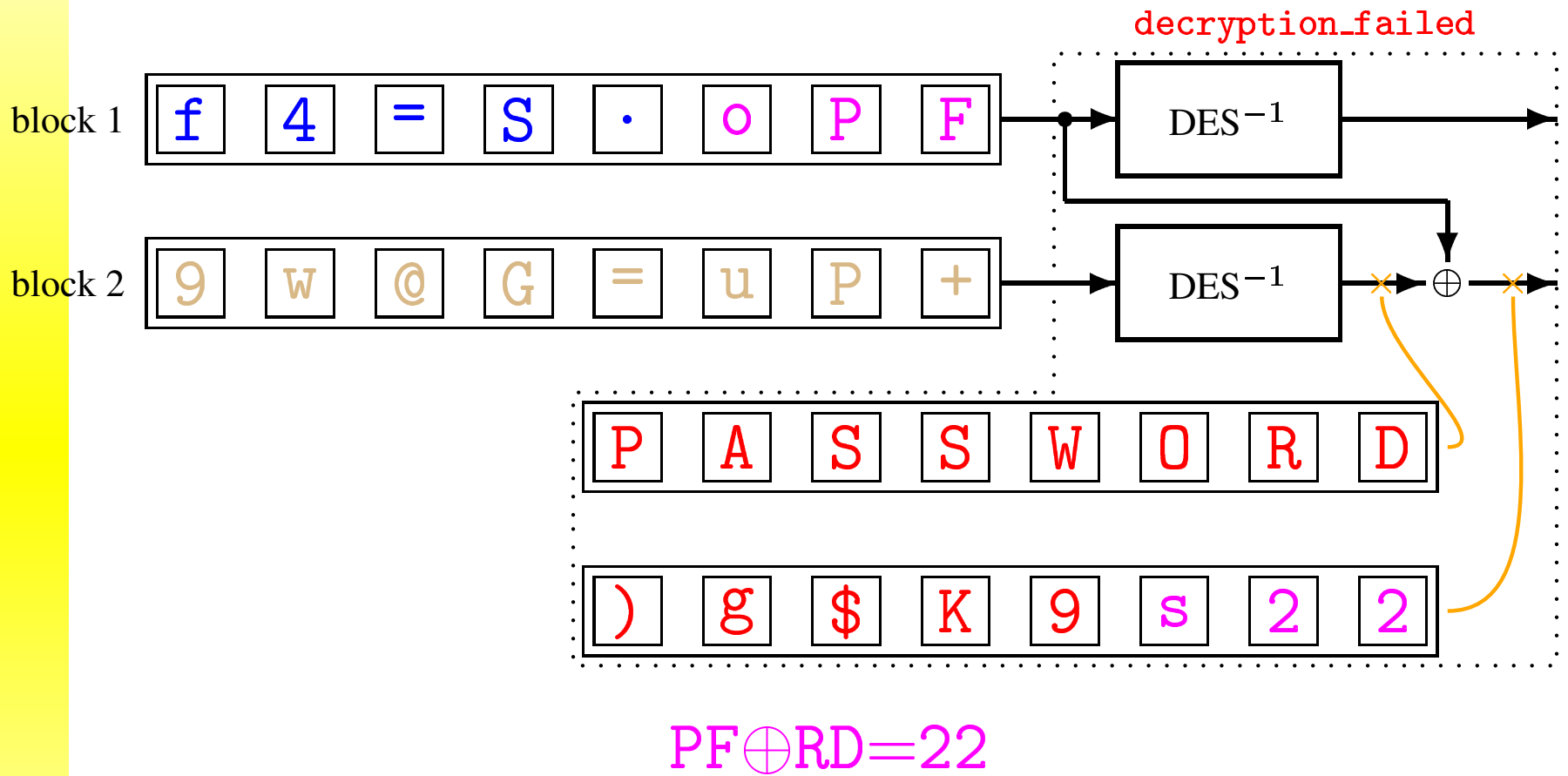
# Side Channel Attack against CBC-PAD



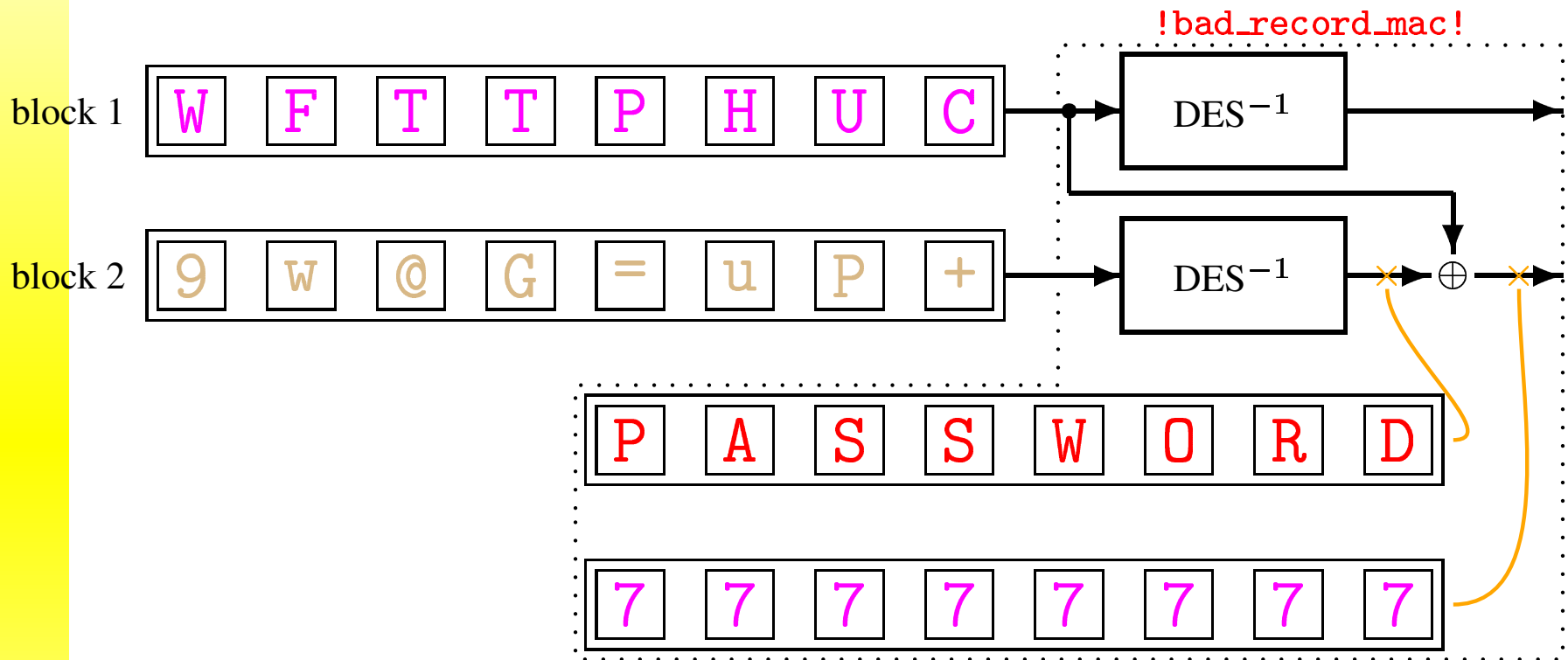
# Side Channel Attack against CBC-PAD



# Side Channel Attack against CBC-PAD



# Side Channel Attack against CBC-PAD



$$\text{WFTTPHUC} \oplus \text{PASSWORD} = 77777777$$

$$\text{RC5}^{-1}(9w@G=uP+) = \text{PASSWORD}$$

## Side Channel Attack against CBC-PAD

---

★ THIS DOES NOT WORK AGAINST TLS:

## Side Channel Attack against CBC-PAD

---

- ★ THIS DOES NOT WORK AGAINST TLS:
- ★ When an error occurs, the session is broken and needs to restart with freshly exchanged keys

## Side Channel Attack against CBC-PAD

---

- ★ THIS DOES NOT WORK AGAINST TLS:
- ★ When an error occurs, the session is broken and needs to restart with freshly exchanged keys
- ★ Error messages in TLS are encrypted

# Timing Attack

# Timing Attack

---

- ★ In order to distinguish between error messages, we perform a **TIMING ATTACK**

# Timing Attack

---

- ★ In order to distinguish between error messages, we perform a TIMING ATTACK
- ★ The idea is that we expect the checking of the MAC to take longer than the checking for the padding

# Timing Attack

---

- ★ In order to distinguish between error messages, we perform a **TIMING ATTACK**
- ★ The idea is that we expect the checking of the MAC to take longer than the checking for the padding
- ★ In order to increase this timing, we construct messages so that they have maximum length

## Timing Attack: Some Notations

---

- ★  $D_W$  (resp.  $D_R$ ) : distribution of the timing answer from the server when there is a padding error (resp. a MAC error).

## Timing Attack: Some Notations

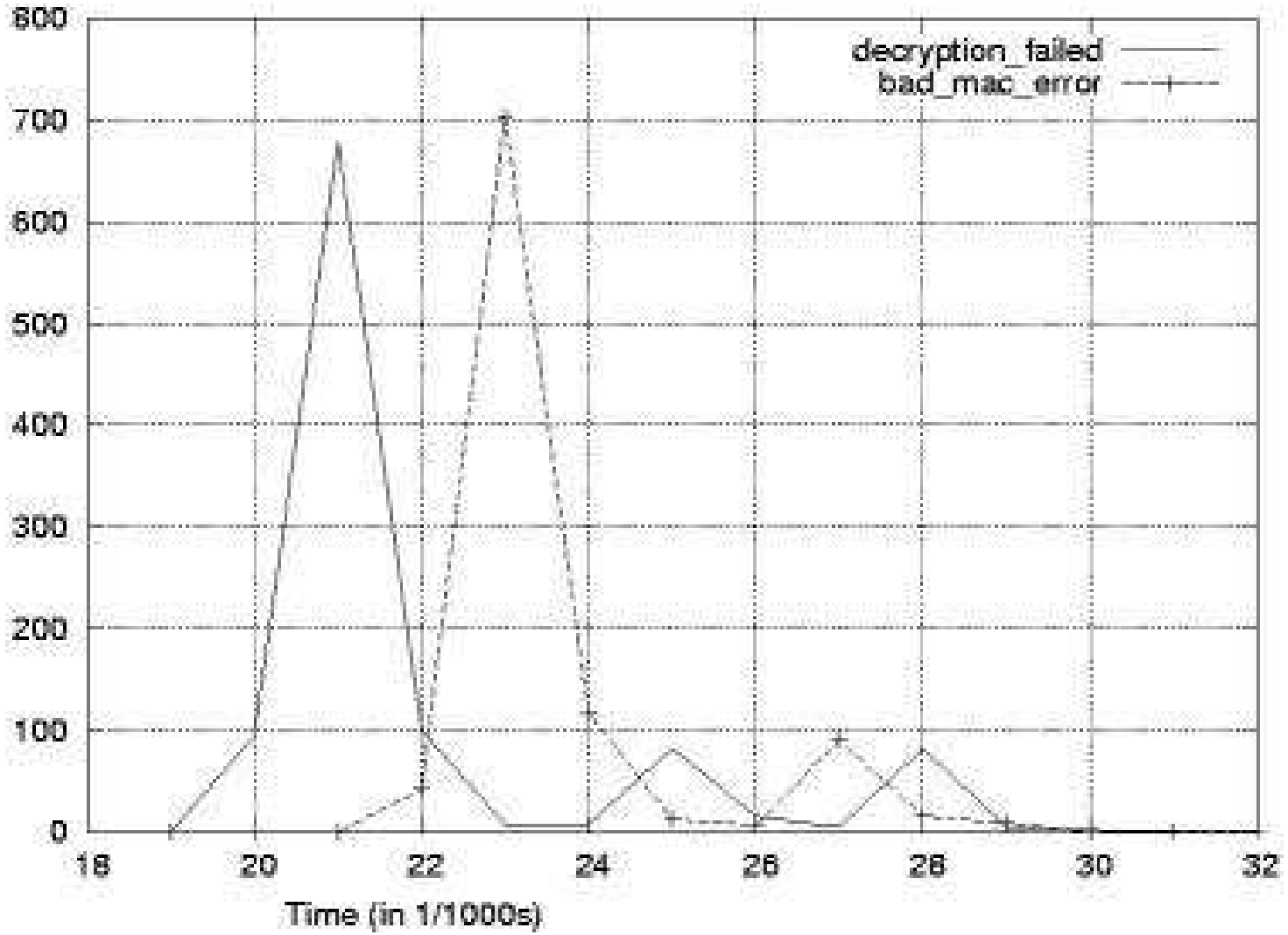
---

- ★  $D_W$  (resp.  $D_R$ ) : distribution of the timing answer from the server when there is a padding error (resp. a MAC error).
- ★  $\mu_W$  (resp.  $\mu_R$ ) : expected values

## Timing Attack: Some Notations

- ★  $D_W$  (resp.  $D_R$ ) : distribution of the timing answer from the server when there is a padding error (resp. a MAC error).
- ★  $\mu_W$  (resp.  $\mu_R$ ) : expected values
- ★  $D_R$  and  $D_W$  approximated by normal distributions with standard deviation  $\sigma$

# Timing Attack: Experimental Values



## Timing Attack: Experimental Values

---

- ★ In our experiments, we have used the following values:

$$\mu_R \approx 23.63ms \quad \mu_W \approx 21.57ms$$

$$\sigma \approx 1.86$$

## Timing Attack: Experimental Values

---

- ★ In our experiments, we have used the following values:

$$\mu_R \approx 23.63ms \quad \mu_W \approx 21.57ms$$

$$\sigma \approx 1.86$$

- ★ Client and attacker on the same LAN

## Timing Attack: Experimental Values

- ★ In our experiments, we have used the following values:

$$\mu_R \approx 23.63ms \quad \mu_W \approx 21.57ms$$

$$\sigma \approx 1.86$$

- ★ Client and attacker on the same LAN
- ★ Server on a different LAN

## Timing Attack: Experimental Values

- ★ In our experiments, we have used the following values:

$$\mu_R \approx 23.63ms \quad \mu_W \approx 21.57ms$$

$$\sigma \approx 1.86$$

- ★ Client and attacker on the same LAN
- ★ Server on a different LAN
- ★ 2 switches and a firewall between the 2 LANs

## Timing Attack: Basic Attack

---

- ★ Query the oracle  $n$  times and collect timings

$T_1, \dots, T_n$

## Timing Attack: Basic Attack

- ★ Query the oracle  $n$  times and collect timings

$T_1, \dots, T_n$

- ★ Accept if

$$\text{ACCEPT : } \frac{T_1 + \dots + T_n}{n} > \tau$$

## Timing Attack: Using Sequential Decision Rules

- ★ By using the theory of hypothesis testing with sequential distinguishers (see Junod, Eurocrypt'03), a more efficient algorithm is obtained with the following STOP and ACCEPT tests.

$$\text{STOP : } T_1 + \dots + T_j - j \frac{\mu_R + \mu_W}{2} \notin [\tau_-, \tau_+]$$

$$\text{ACCEPT : } T_1 + \dots + T_j - j \frac{\mu_R + \mu_W}{2} > \tau_+$$

where  $\tau_-$  and  $\tau_+$  are two given thresholds.



# Multisession Attack

# Multisession Attack

---

- ★ Here we solve the problem of the broken sessions in performing the attack on TLS

# Multisession Attack

---

- ★ Here we solve the problem of the broken sessions in performing the attack on TLS
- ★ Assume that each session includes a critical plaintext block  $x$  which is always the same (e.g. password)

# Multisession Attack

- ★ Here we solve the problem of the broken sessions in performing the attack on TLS
- ★ Assume that each session includes a critical plaintext block  $x$  which is always the same (e.g. password)
- ★ Assume we intercept the corresponding ciphertext  $y = \text{ENC}(x \oplus y')$ . (Here  $y'$  is the previous ciphertext block following the CBC mode.)

# Multisession Attack

- ★ Here we solve the problem of the broken sessions in performing the attack on TLS
- ★ Assume that each session includes a critical plaintext block  $x$  which is always the same (e.g. password)
- ★ Assume we intercept the corresponding ciphertext  $y = \text{ENC}(x \oplus y')$ . (Here  $y'$  is the previous ciphertext block following the CBC mode.)
- ★ The target  $x$  is constant in every session, but  $y$  and  $y'$  depend on the session.

# Multisession Attack

---

- ★ We have performed two types of attacks:

# Multisession Attack

---

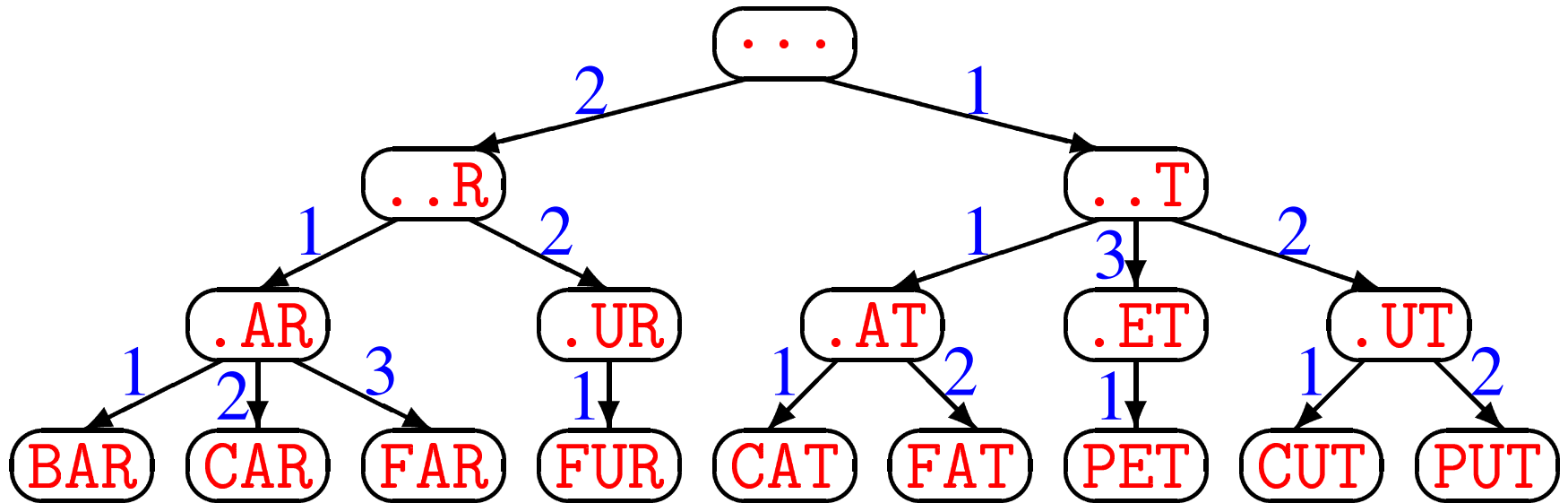
- ★ We have performed two types of attacks:
- ★ Case 1: the target block is uniformly distributed in an alphabet of size  $|Z|$

# Multisession Attack

---

- ★ We have performed two types of attacks:
- ★ Case 1: the target block is uniformly distributed in an alphabet of size  $|Z|$
- ★ Case 2: we know the a priori distribution of the characters of the block we want to decrypt (dictionary attack)

# Multisession Attack : Dictionary Case



$$C_0 = \sum_{i=1}^b \sum_{c_i, \dots, c_1} \Pr[c_i, \dots, c_1] N(c_i \dots c_1)$$

# Multisession Attack

$$C \approx \frac{2\sigma^2}{(\mu_R - \mu_W)^2} \left( K - C_0 \log \log \frac{1}{p} \right)$$

where

$$K = C_0 \log C_0 + (C_0 - 2b) (\log b - \log(C_0 - b))$$

Note :  $C_0$  is the average complexity to decrypt a block when asking only one question to the oracle and  $p$  is the probability of success

# Multisession Attack

Dictionary,  $C_0 = 31$

$p$	0.5	0.6	0.7	0.8	0.9	0.99
$C$	166	181	199	223	261	380

Uniform distribution,  $|Z| = 256$ ,  $C_0 = 1028$

$p$	0.5	0.6	0.7	0.8	0.9	0.99
$C$	4239	4750	5353	6139	7397	11335



# Experiments and Discussions

# Password Interception

---

- ★ IMAP client: Outlook Express 6.x from Microsoft under Windows XP

# Password Interception

---

- ★ IMAP client: Outlook Express 6.x from Microsoft under Windows XP
- ★ IMAP Rev 4 server (Taken from Pine, Washington University)

# Password Interception

---

- ★ IMAP client: Outlook Express 6.x from Microsoft under Windows XP
- ★ IMAP Rev 4 server (Taken from Pine, Washington University)
- ★ Outlook checks (by default) for messages automatically every 5 minutes each folder created on the IMAP user account

# Password Interception

- ★ IMAP client: Outlook Express 6.x from Microsoft under Windows XP
- ★ IMAP Rev 4 server (Taken from Pine, Washington University)
- ★ Outlook checks (by default) for messages automatically every 5 minutes each folder created on the IMAP user account
- ★ E.g. five folders (in, out, trash, read, and draft)  
→ 60 sessions every hour

# Multisession Attack

Dictionary,  $C_0 = 31$

$p$	0.5	0.6	0.7	0.8	0.9	0.99
$C$	166	181	199	223	261	380

Uniform distribution,  $|Z| = 256$ ,  $C_0 = 1028$

$p$	0.5	0.6	0.7	0.8	0.9	0.99
$C$	4239	4750	5353	6139	7397	11335

# Cipher Problem

---

- ★ Outlook uses the RC4\_MD5 algorithm by default (despite RFC2246 and RFC2595 suggest that 3DES\_EDE\_CBC\_SHA should be supported by default).

# Cipher Problem

---

- ★ Outlook uses the RC4\_MD5 algorithm by default (despite RFC2246 and RFC2595 suggest that 3DES\_EDE\_CBC\_SHA should be supported by default).
- ★ We had to force the IMAP server to only offer block ciphers in CBC mode.

# Cipher Problem

---

- ★ Outlook uses the RC4\_MD5 algorithm by default (despite RFC2246 and RFC2595 suggest that 3DES\_EDE\_CBC\_SHA should be supported by default).
- ★ We had to force the IMAP server to only offer block ciphers in CBC mode.
- ★ Other applications (e.g. stunnel) use block ciphers by default.

## Conditions for a Successful Attack

---

- ★ A critical piece of information is repeatedly encrypted at a predictable place.

## Conditions for a Successful Attack

---

- ★ A critical piece of information is repeatedly encrypted at a predictable place.
- ★ A block cipher in CBC mode is chosen.

## Conditions for a Successful Attack

---

- ★ A critical piece of information is repeatedly encrypted at a predictable place.
- ★ A block cipher in CBC mode is chosen.
- ★ The attacker can sit in the middle and perform active attacks.

## Conditions for a Successful Attack

---

- ★ A critical piece of information is repeatedly encrypted at a predictable place.
- ★ A block cipher in CBC mode is chosen.
- ★ The attacker can sit in the middle and perform active attacks.
- ★ The attacker can distinguish time differences between two types of errors.

# Countermeasures

---

- ★ A countermeasure for TLS has been implemented in OpenSSL 0.9.6d and following versions:  
only the `bad_mac_error` error message is sent when an incorrect padding or an incorrect MAC are detected.

# Countermeasures

---

- ★ A countermeasure for TLS has been implemented in OpenSSL 0.9.6d and following versions:  
only the `bad_mac_error` error message is sent when an incorrect padding or an incorrect MAC are detected.
- ★ This countermeasure is not enough because of timing attacks.

# Countermeasures

- ★ A countermeasure for TLS has been implemented in OpenSSL 0.9.6d and following versions:  
only the `bad_mac_error` error message is sent when an incorrect padding or an incorrect MAC are detected.
- ★ This countermeasure is not enough because of timing attacks.
- ★ A new countermeasure was implemented in OpenSSL 0.9.6i:  
we always check a MAC even if the padding is not correct.

# Countermeasures

- ★ A countermeasure for TLS has been implemented in OpenSSL 0.9.6d and following versions:  
only the `bad_mac_error` error message is sent when an incorrect padding or an incorrect MAC are detected.
- ★ This countermeasure is not enough because of timing attacks.
- ★ A new countermeasure was implemented in OpenSSL 0.9.6i:  
we always check a MAC even if the padding is not correct.
- ★ Other possible countermeasure:  
invert the padding and the MAC!

# Conclusions

---

- ★ TLS implements a nice protocol for secure tunnel set up

# Conclusions

---

- ★ TLS implements a nice protocol for secure tunnel set up
- ★ Despite the TLS maturity and popularity, it is not flaw free

# Conclusions

---

- ★ TLS implements a nice protocol for secure tunnel set up
- ★ Despite the TLS maturity and popularity, it is not flaw free
- ★ We can make timing attacks over a LAN

# Conclusions

---

- ★ TLS implements a nice protocol for secure tunnel set up
- ★ Despite the TLS maturity and popularity, it is not flaw free
- ★ We can make timing attacks over a LAN
- ★ The order MAC-PAD-Encrypt should be reconsidered

# Thank You

---

Special thank you to the IACR and Greg Rose for having made it possible for me to be here at Crypto.