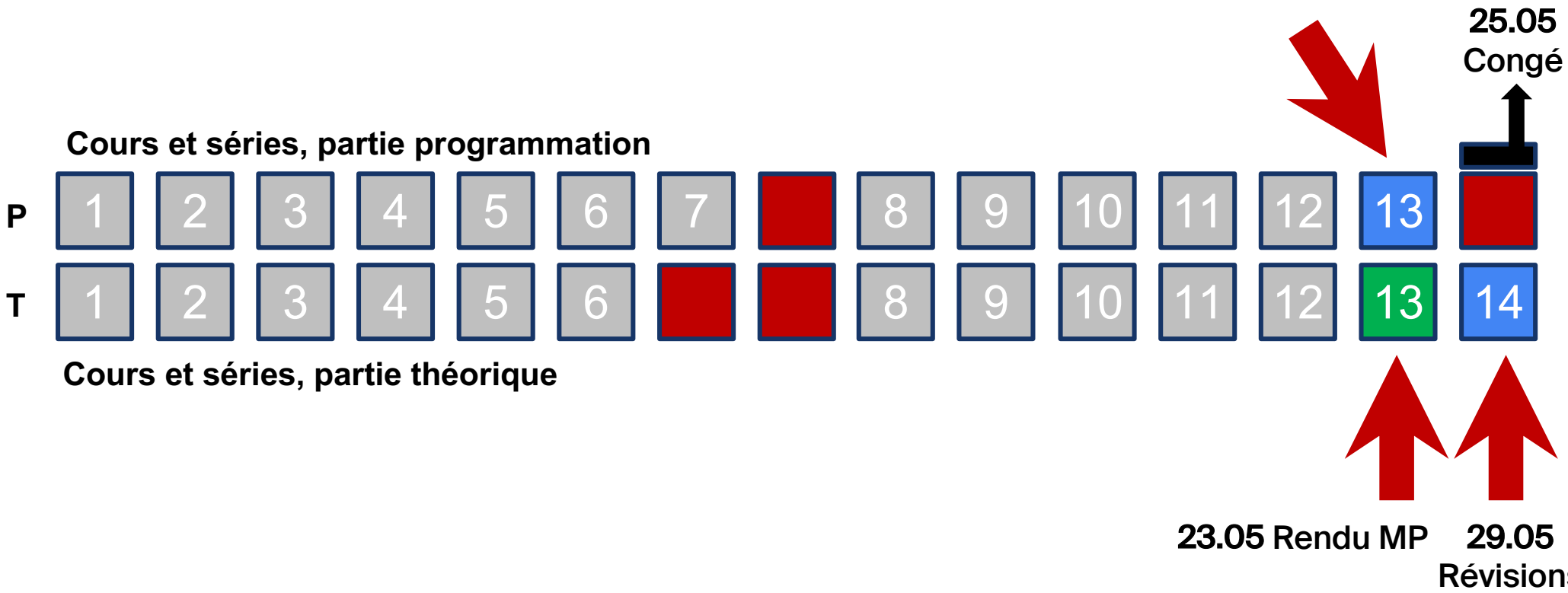
Information, Calcul et Communication

CS-119(k) ICC – Programmation
Semaine 13

Rafael Pires
rafael.pires@epfl.ch

Lausanne, 18.05.2026

Planning



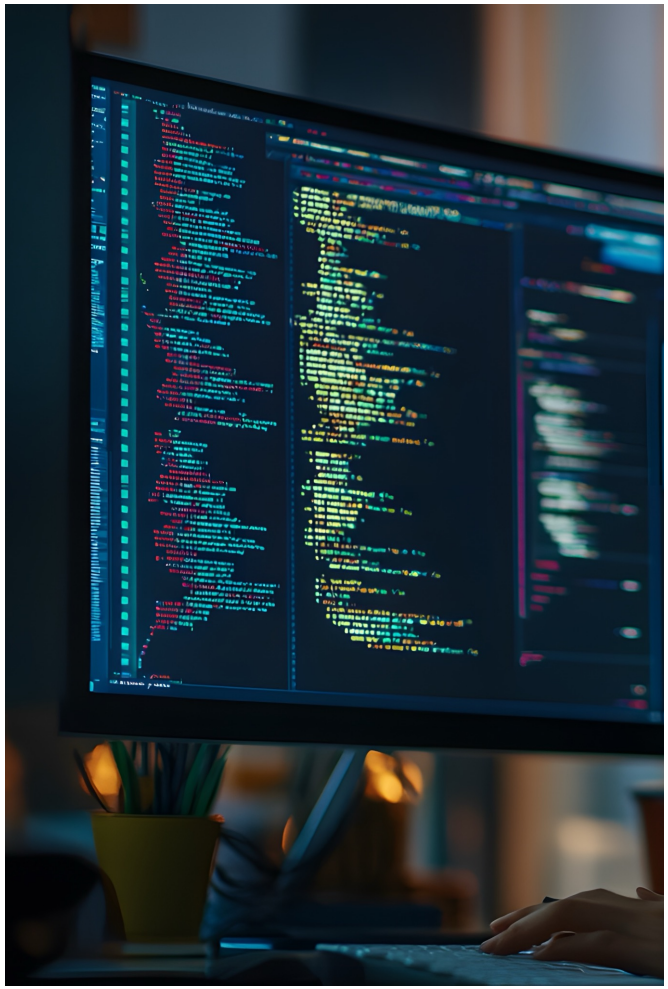
26.06 9h15 Examen **P (1/3)** et **T (2/3)** (**65%**)

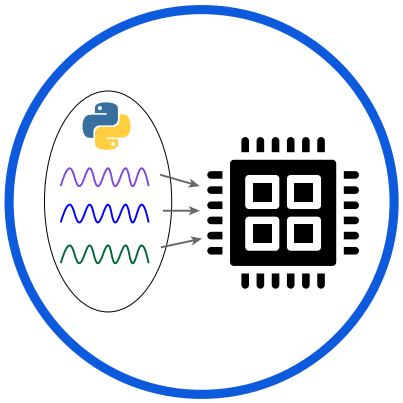
Évaluation approfondie

- Dès aujourd'hui jusqu'au dimanche 7 juin
  - Connectez-vous à Moodle et restez sur la page d'accueil (tableau de bord, pas la page du cours)
  - Cliquer sur la flèche en haut à droite de l'écran qui fera apparaître un bloc contenant la tuile intitulée « Évaluation approfondie »

- Donnez un retour sur le cours
  - Feedback anonyme
  - Dites ce qui vous a plu, ce qui vous a déplu, et vos suggestions d'amélioration
  - Soyez constructifs : c'est votre opportunité d'aider à améliorer ce cours à l'avenir

Précédemment, dans... ICC-P





- Les threads servent à exécuter plusieurs morceaux de code « à la fois »
- On peut avoir des problèmes lorsque ces threads changent des variables partagées
- Les locks protègent l'accès aux variables partagées
- Un deadlock peut survenir si plusieurs threads s'attendent les uns les autres

Cryptographie

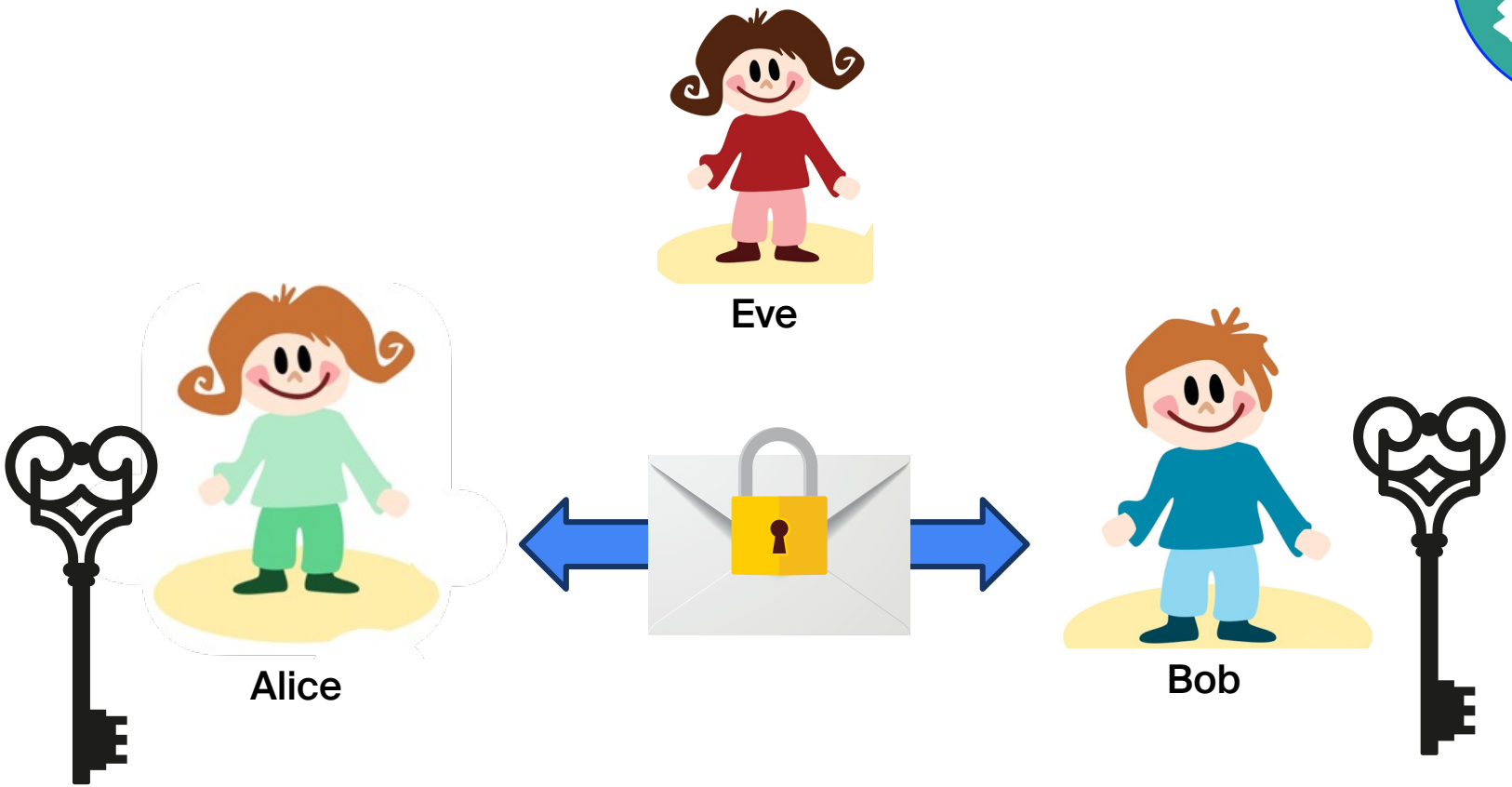
Cryptographie

Eve

Alice

Bob

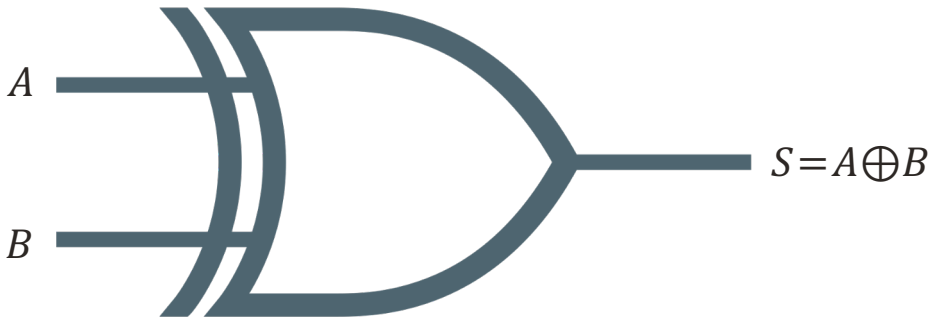
Cryptographie à clé secrète (symétrique)



Cryptographie à clé secrète (symétrique)

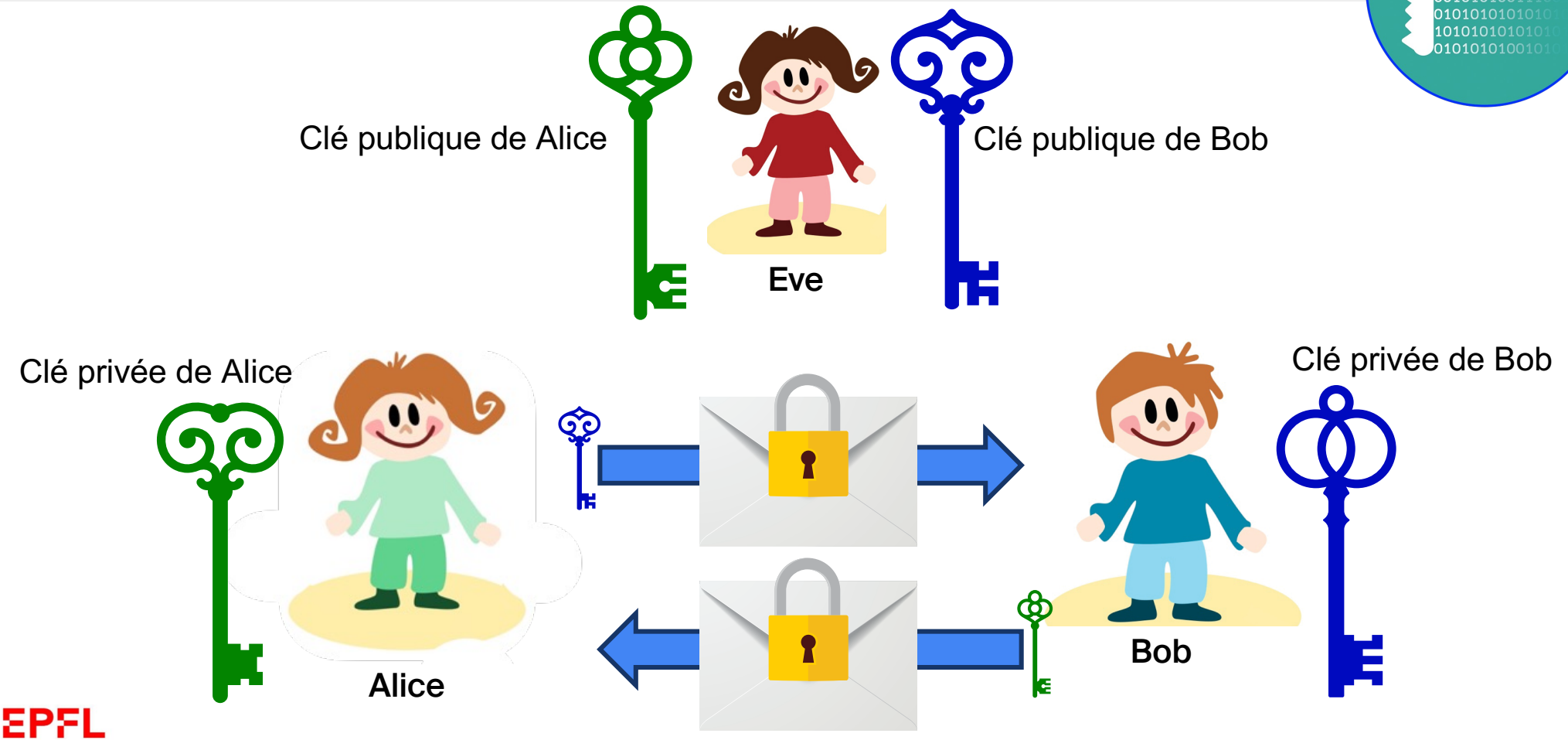
```python
def xor_encrypt_decrypt(message: str, key: int) -> str:
    return ''.join(chr(ord(c) ^ key) for c in message)

# Message original
message = "HELLO"
key = 42 # Clé secrète partagée

# Chiffrement
ciphertext = xor_encrypt_decrypt(message, key)
print("Chiffré :", ciphertext)

# Déchiffrement
decrypted = xor_encrypt_decrypt(ciphertext, key)
print("Déchiffré :", decrypted)
```

ICC-T 08 : La porte OU Exclusif (XOR)



A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Cryptographie à clé publique (asymétrique)



Cryptographie à clé publique (asymétrique)

```python
def mod_exp(base, exp, mod):
  result = 1
  for _ in range(exp):
    result = (result * base) % mod
  return result

def pgdc(a, b):
  while b:
    a, b = b, a % b
  return a

def inverse_mod(e, phi):
  for d in range(1, phi):
    if (e * d) % phi == 1:
      return d
  raise ValueError("Pas d'inverse")
```

```python
# Paramètres RSA simplifiés (petits nombres !)
p, q = 3, 11
n = p * q                  # module
phi = (p - 1) * (q - 1) # indicatrice d'Euler : phi(n)

# Choix de e : 1 < e < phi et pgdc(e, phi) = 1
e = 3
assert 1 < e < phi and pgdc(e, phi) == 1, "e invalide"

# Calcul de d : inverse modulaire de e modulo phi
d = inverse_mod(e, phi)

# Message = un entier < n
message = 4

# Chiffrement (avec la clé publique)
ciphertext = mod_exp(message, e, n)
print("Chiffré :", ciphertext)

# Déchiffrement (avec la clé privée)
decrypted = mod_exp(ciphertext, d, n)
print("Déchiffré :", decrypted)
```

Résumé Semaine 13 – ICC-P

- Cryptographie symétrique : une seule clé secrète partagée
- Cryptographie asymétrique :
  - une clé publique pour chiffrer
  - une clé privée pour déchiffrer
- Plus la clé est longue, plus une attaque par force brute est lente
- Révisions

rafael.pires@epfl.ch

Merci