Chiffrement asymétrique//RSA

Christophe Viroulaud

Terminale NSI

Problématique

Le chiffrement asymétrique de Diffie-Hellman permet d'échanger des clés via un canal non sûr mais ne gère pas les problèmes liés à l'authentification des interlocuteurs.

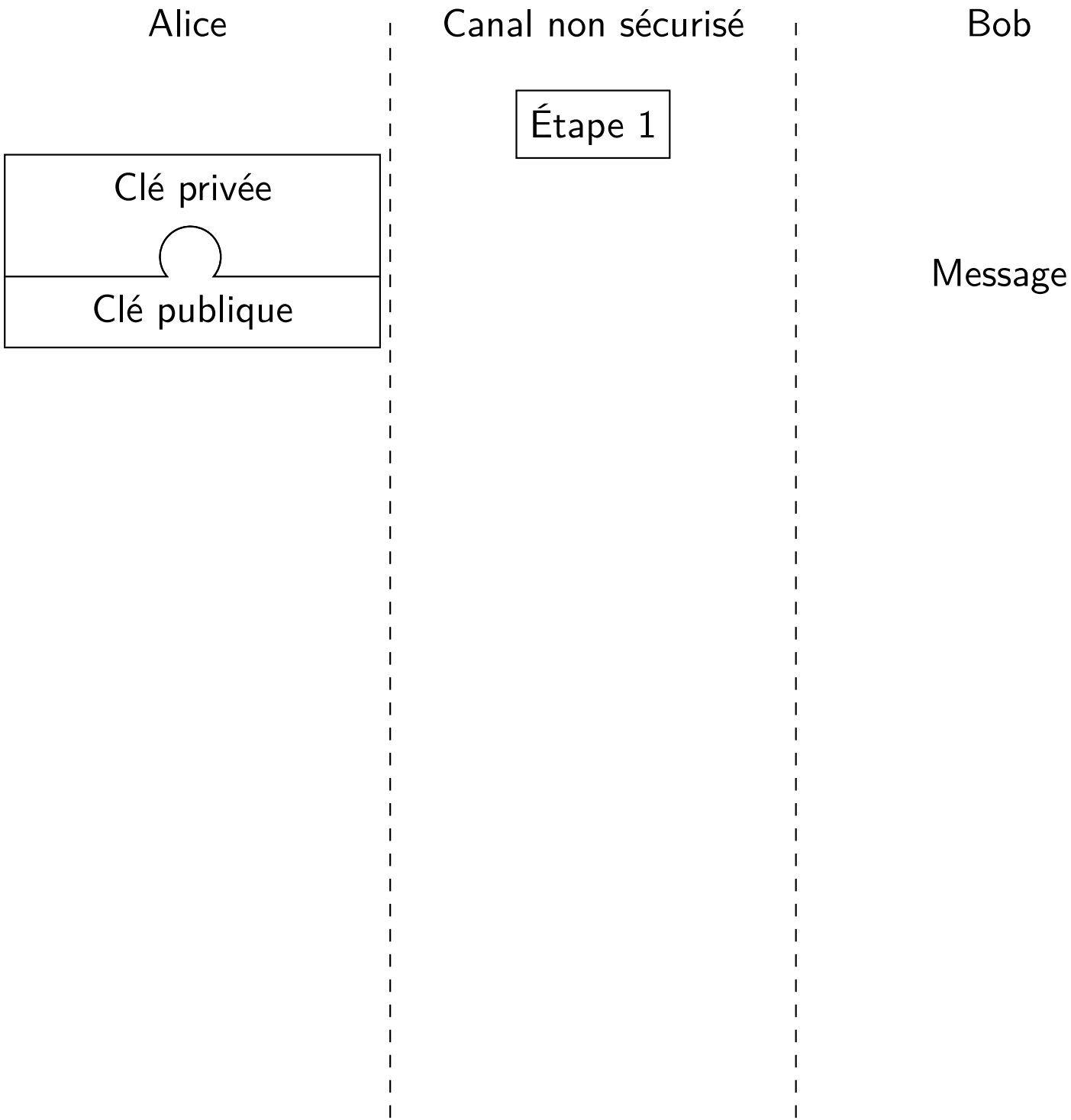
Comment authentifier avec certitude les participants ?

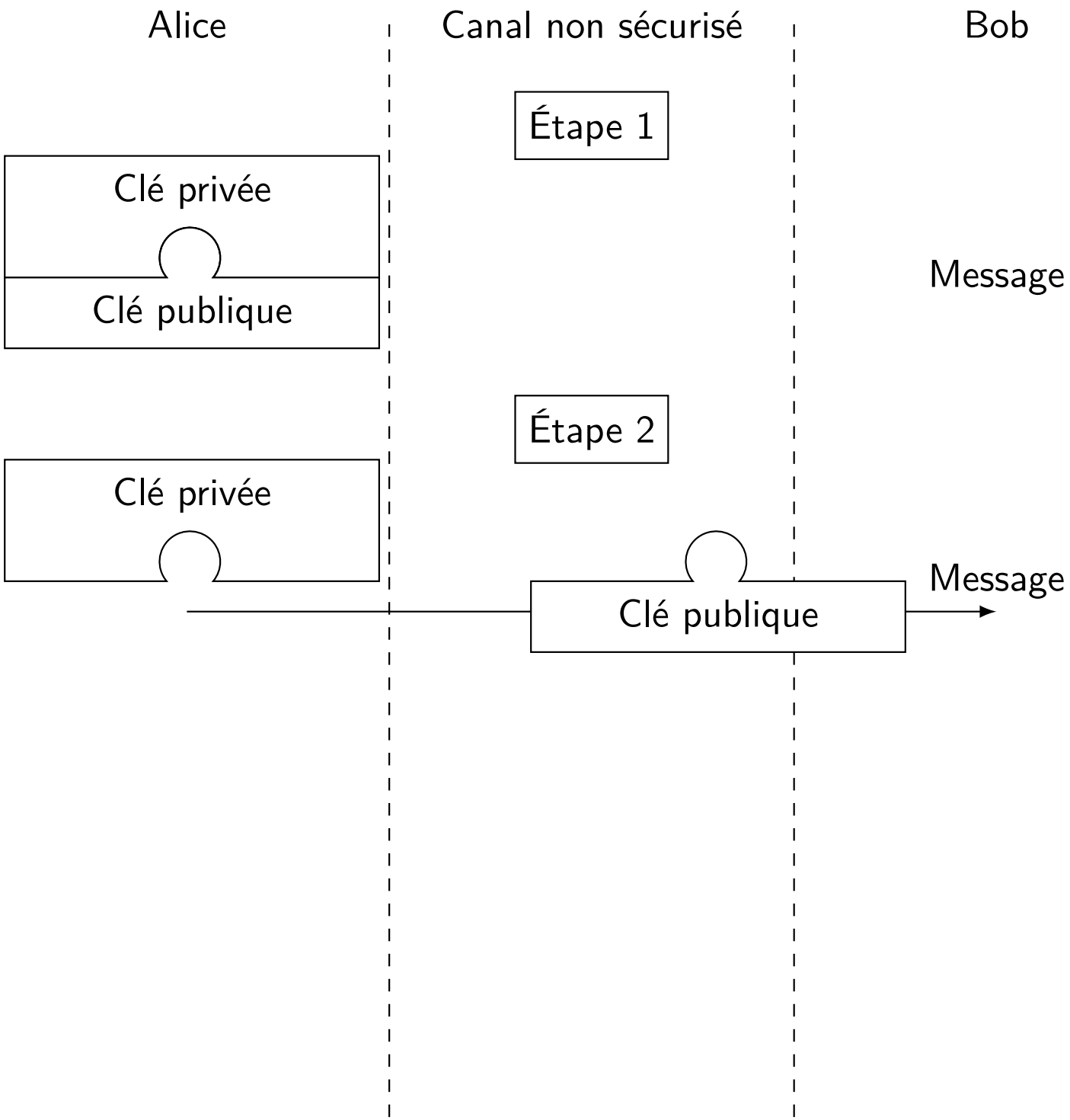
Chiffrement RSA

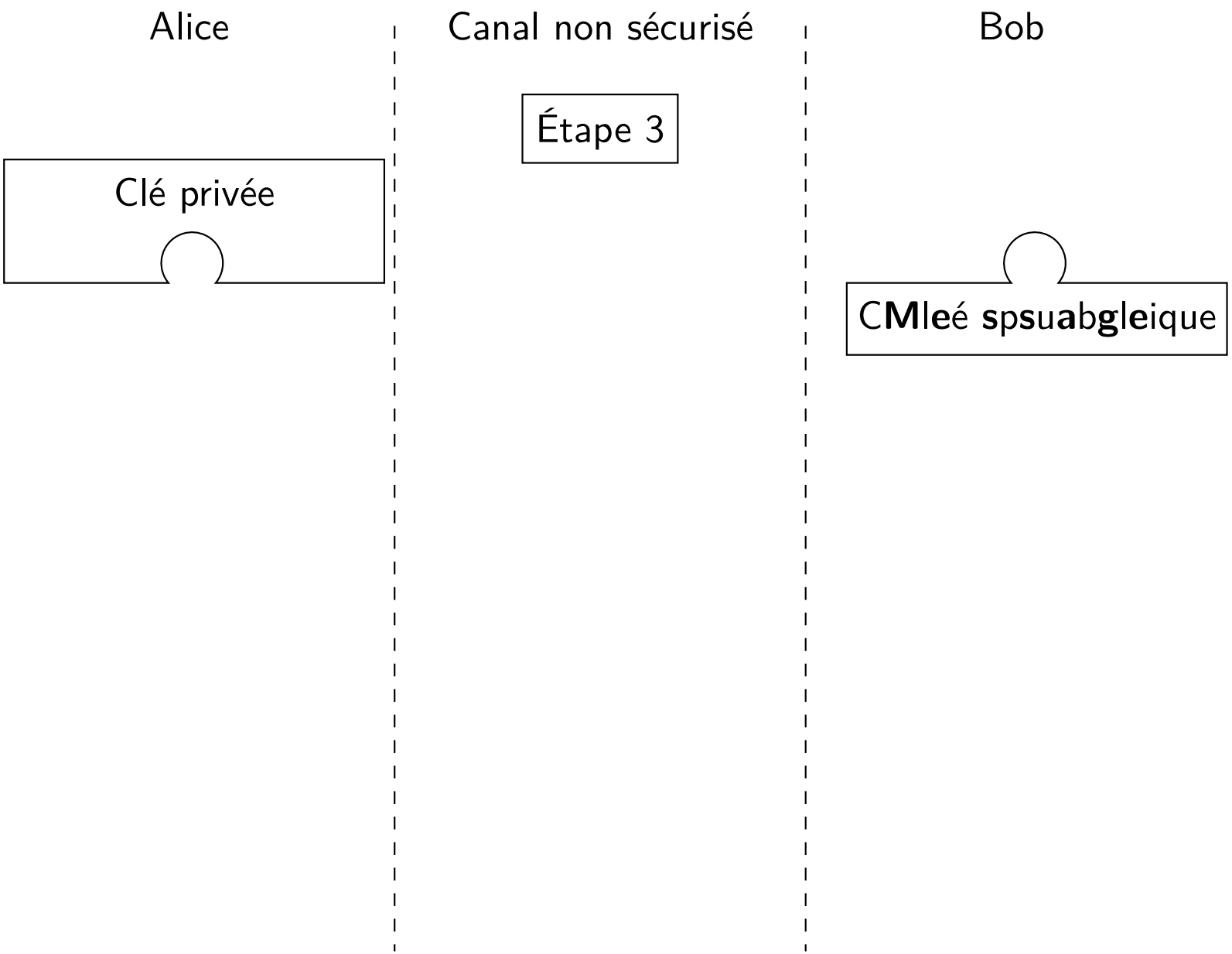
- 1977 : Ron Rivest, Adi Shamir et Len Adleman.
- breveté en 1983 ; expiration du brevet en 2000.
- *fonctions à sens unique* (comme Diffie-Hellman)
- une paire de clés publique et privée.

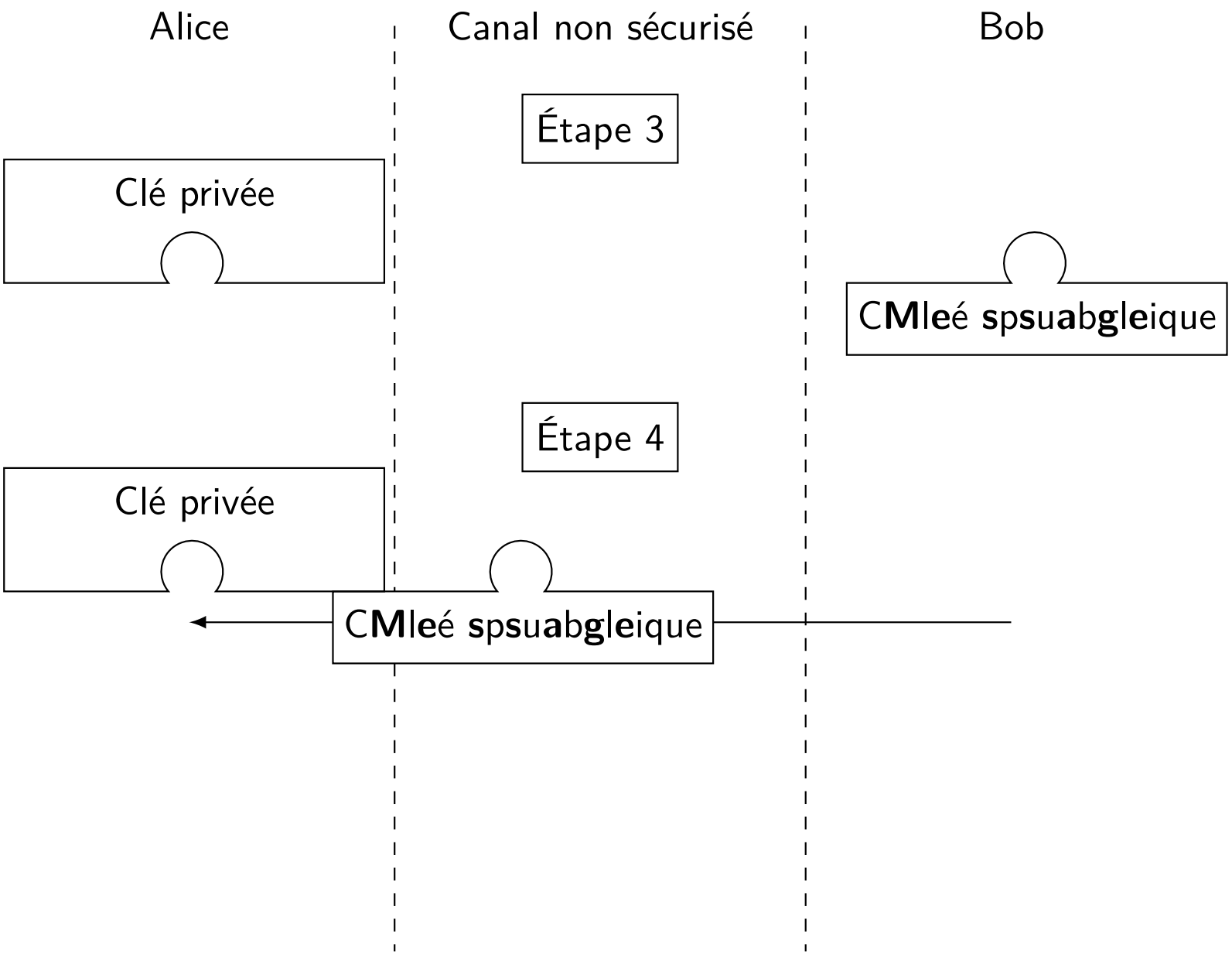
À retenir

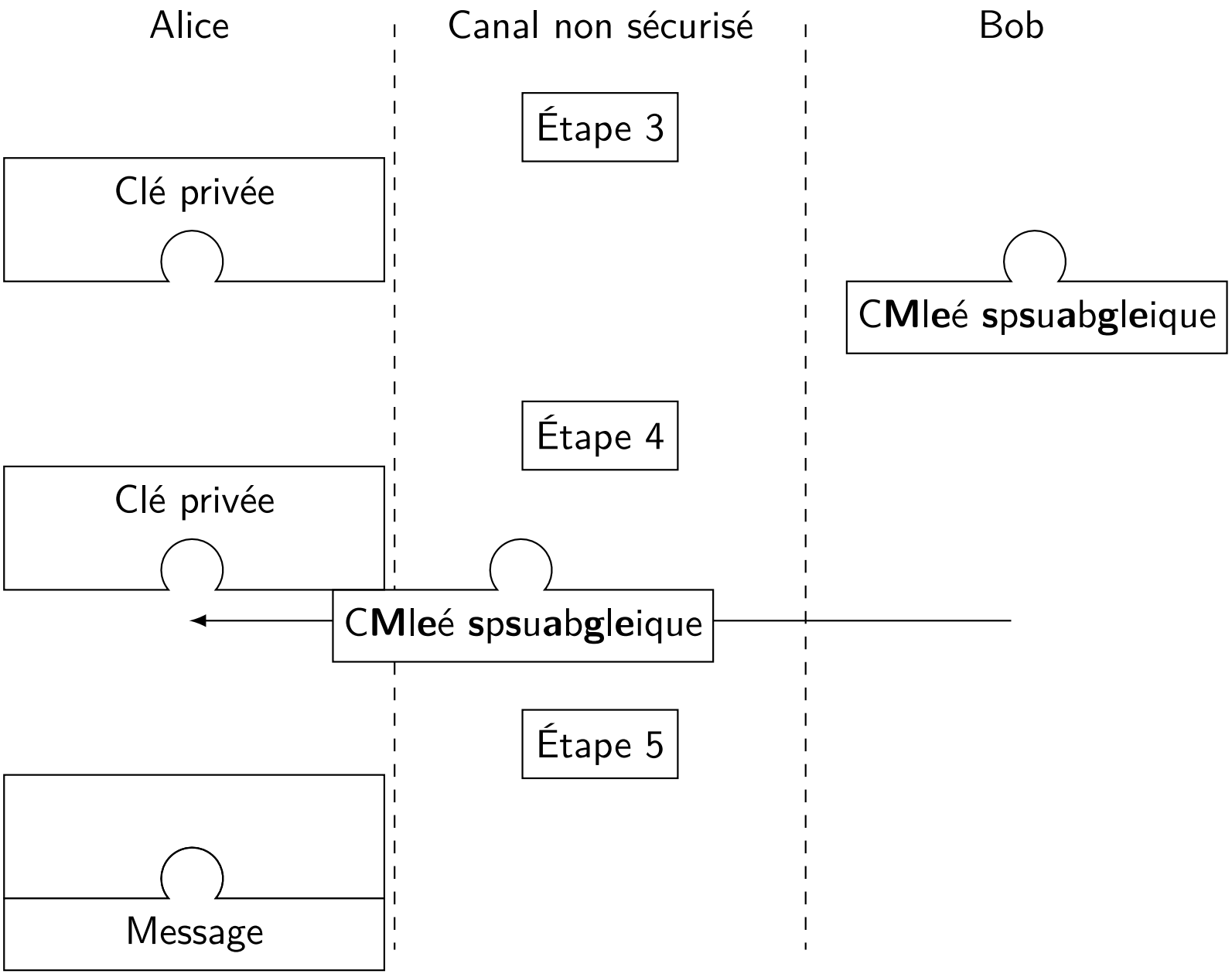
$$K_{priv}(K_{pub}(m)) = K_{pub}(K_{priv}(m)) = m$$


Alice
Canal non sécurisé
Bob
Étape 1
Clé privée
Clé publique
Message


Alice
Canal non sécurisé
Bob
Étape 1
Clé privée
Clé publique
Message
Étape 2
Clé privée
Message
Clé publique







Mathématiquement, la fonction respecte les règles suivantes :

- Il est impossible de deviner la clé privée en connaissant la clé publique.
- Il est impossible de deviner le message avec une seule des deux clés.

kid RSA

La création des clés suit un algorithme mathématique complexe. L'université de Rhode Island a produit une version simplifiée (à utilisation pédagogique) pour simuler le protocole.

Activité 1 :

1. Découvrir l'algorithme *kidrsa* sur la page https://tinyurl.com/rsakid
2. Écrire la fonction **`creer_nombre(a: int, b: int, a1: int, b1: int)`** $\rightarrow$ **`dict`** qui renvoie un dictionnaire contenant les clés privée et publique. Chaque clé sera un tuple.
3. Écrire la fonction **`chiffrer(message: int, publique: tuple)`** $\rightarrow$ **`int`** qui encode *message* avec la clé publique (e, n).
4. Écrire la fonction **`dechiffrer(message_chiffre: int, privee: tuple)`** $\rightarrow$ **`int`** qui déchiffre *message_chiffre* avec la clé privée (d, n).
5. Tester l'algorithme de chiffrage avec un entier (inférieur à n).

Correction

```
1  def creer_nombre(a: int, b: int, a1: int, b1: int) -> dict
       :
2      """
3      crée un couple clé privée/publique
4      Returns:
5          dict: {"publique" : (e, n), "privee":(d, n)}
6      """
7      M = a*b-1
8      e = a1*M+a
9      d = b1*M+b
10     n = (e*d)//M
11     return {"publique": (e, n), "privee": (d, n)}
```

Code 1 – Alice crée ses clés

Correction

```
1  def chiffrer(message: int, publique: tuple) -> int:
2      """
3      Args:
4          message (int)
5          publique (tuple): (e, n)
6
7      Returns:
8          int: message chiffré
9      """
10     return (publique[0]*message) % publique[1]
```

Code 2 – Bob chiffre son message avec la clé publiqe d'Alice

Correction

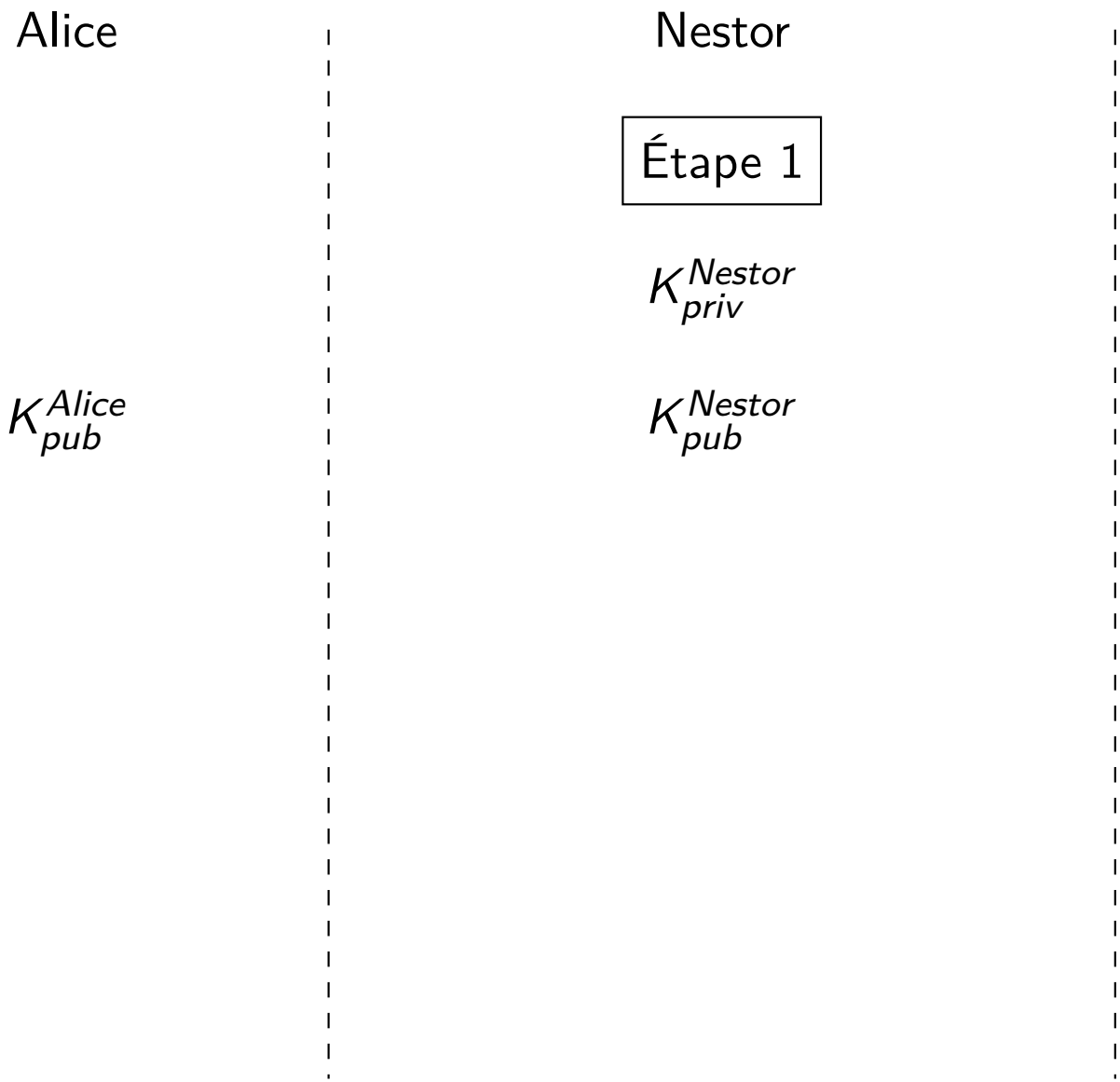
```
1  def dechiffrer(message_secret: int, privee: tuple) -> int:
2      """
3      Args:
4          message_secret (int)
5          privee (tuple): (d, n)
6
7      Returns:
8          int: message déchiffré
9      """
10     return (privee[0]*message_secret) % privee[1]
```

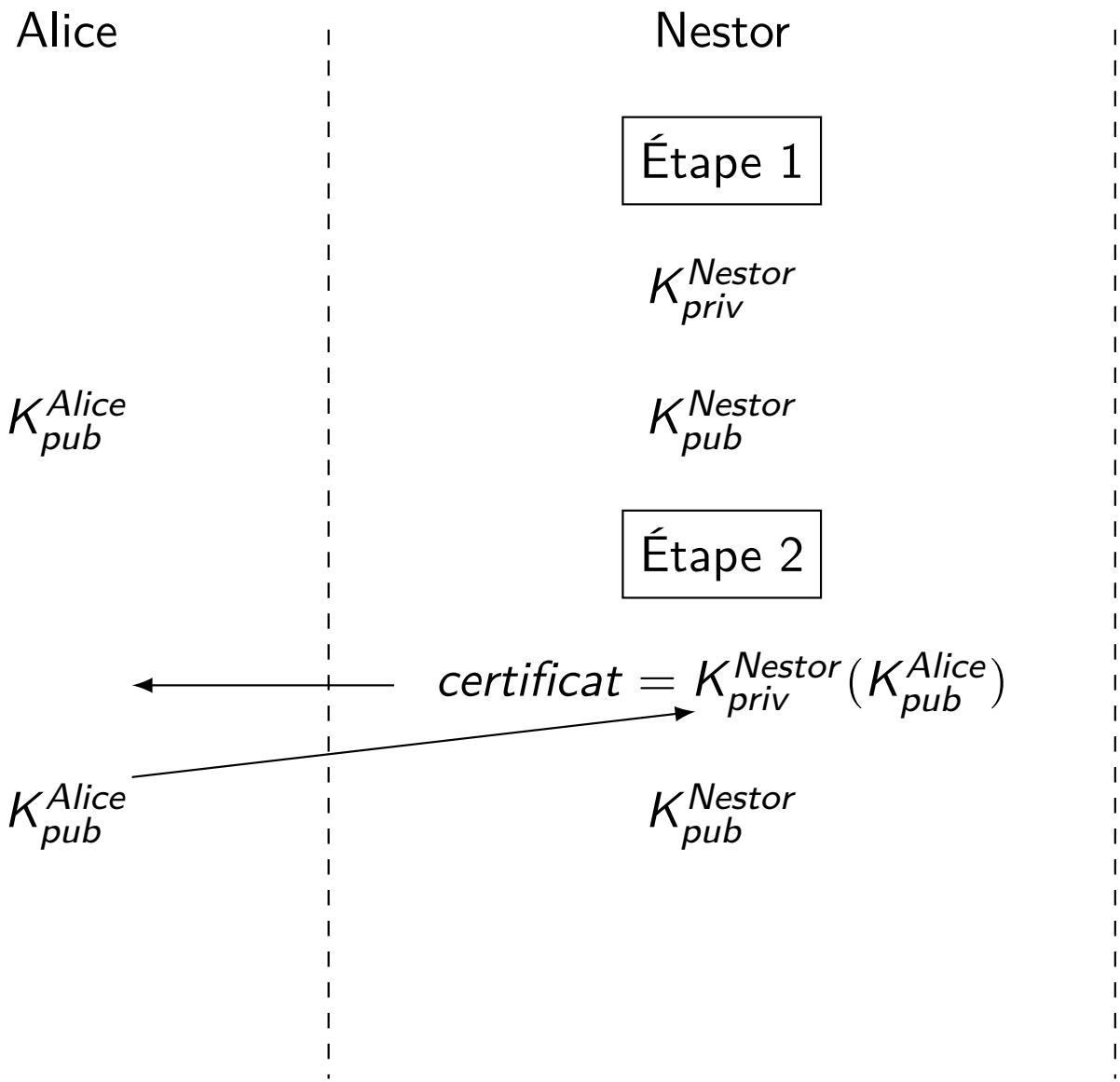
Code 3 – Alice déchiffre le message de Bob avec sa clé privée

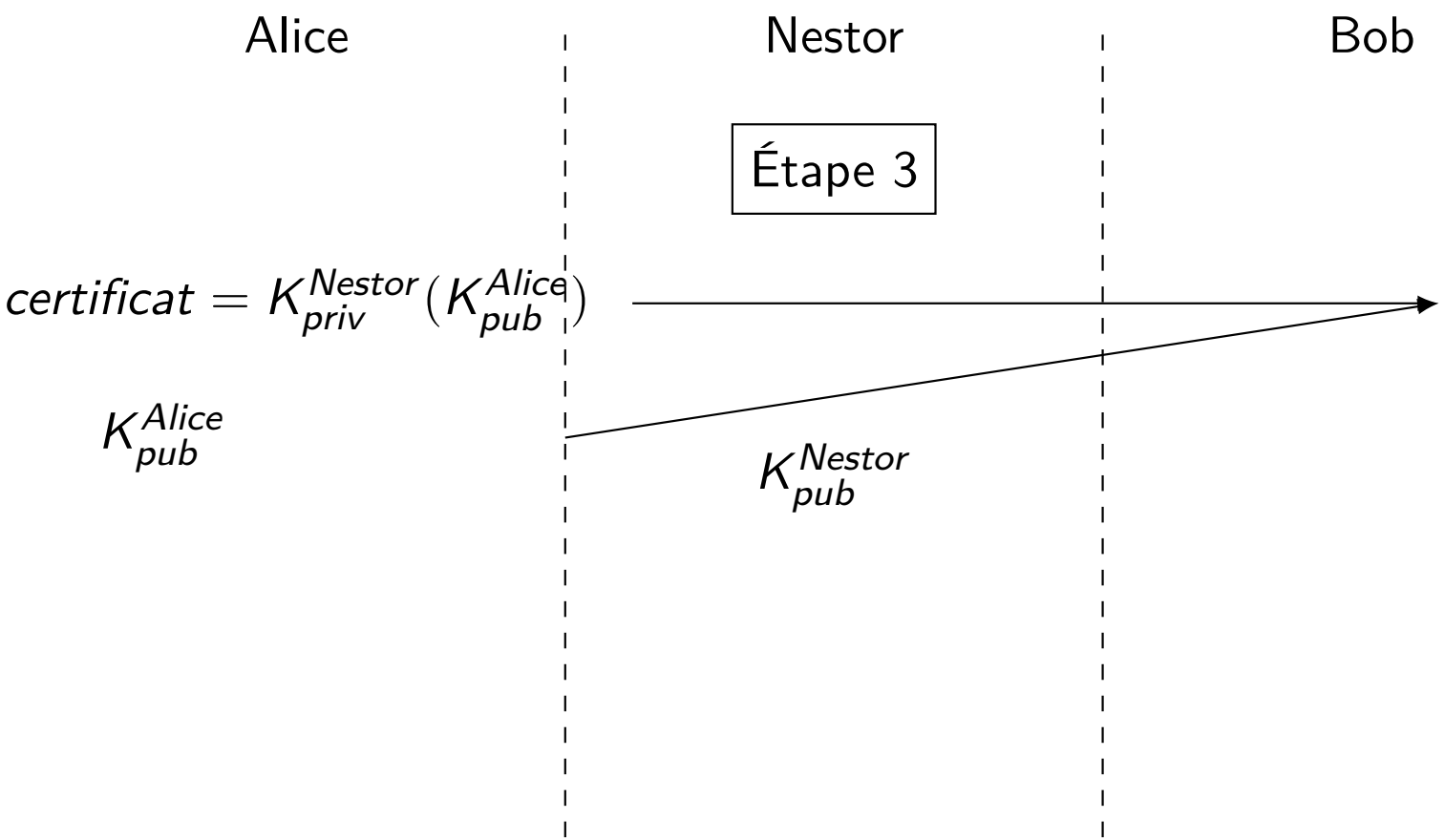
Pour l'instant, l'algorithme RSA ne fait rien de plus que celui de Diffie-Hellman. Nous n'avons toujours pas réglé le problème de l'authentification.

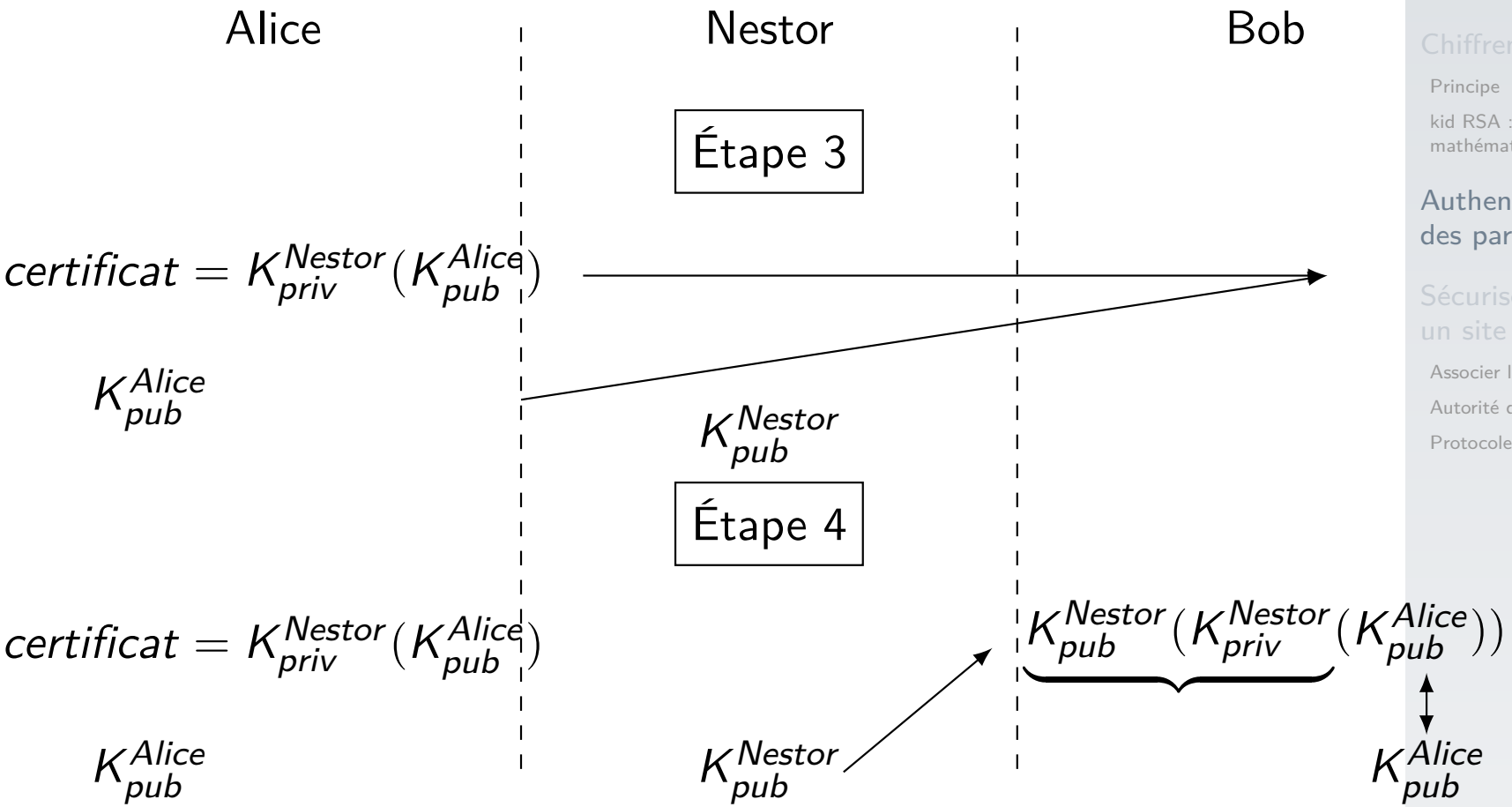
Authentification

Il faut qu'un **tiers de confiance** (Nestor) intervienne.


Alice
Nestor
Étape 1
K^{Nestor}_{priv}
K^{Alice}_{pub}
K^{Nestor}_{pub}







Sécuriser l'accès à un site web

L'algorithme RSA permet de sécuriser les données mais également d'authentifier les participants. Il semble être le candidat idéal pour effectuer toutes ces tâches. Cependant, il est très coûteux en temps de calcul.

À retenir

On mettra à profit les avantages de chaque type de chiffrement :

- Le chiffrement symétrique, rapide, sera utilisé pour coder les données avec *clé de chiffrement*.
- Le chiffrement asymétrique, permettant d'authentifier les participants, sera utilisé pour transmettre la clé de chiffrement symétrique.

Autorité de certification

Une autorité de certification peut être :

- un état,
- une entreprise spécialisée,
- une association à but non lucratif (Let's Encrypt).

Les navigateurs possèdent une copie des clés publiques de ces autorités de certification.

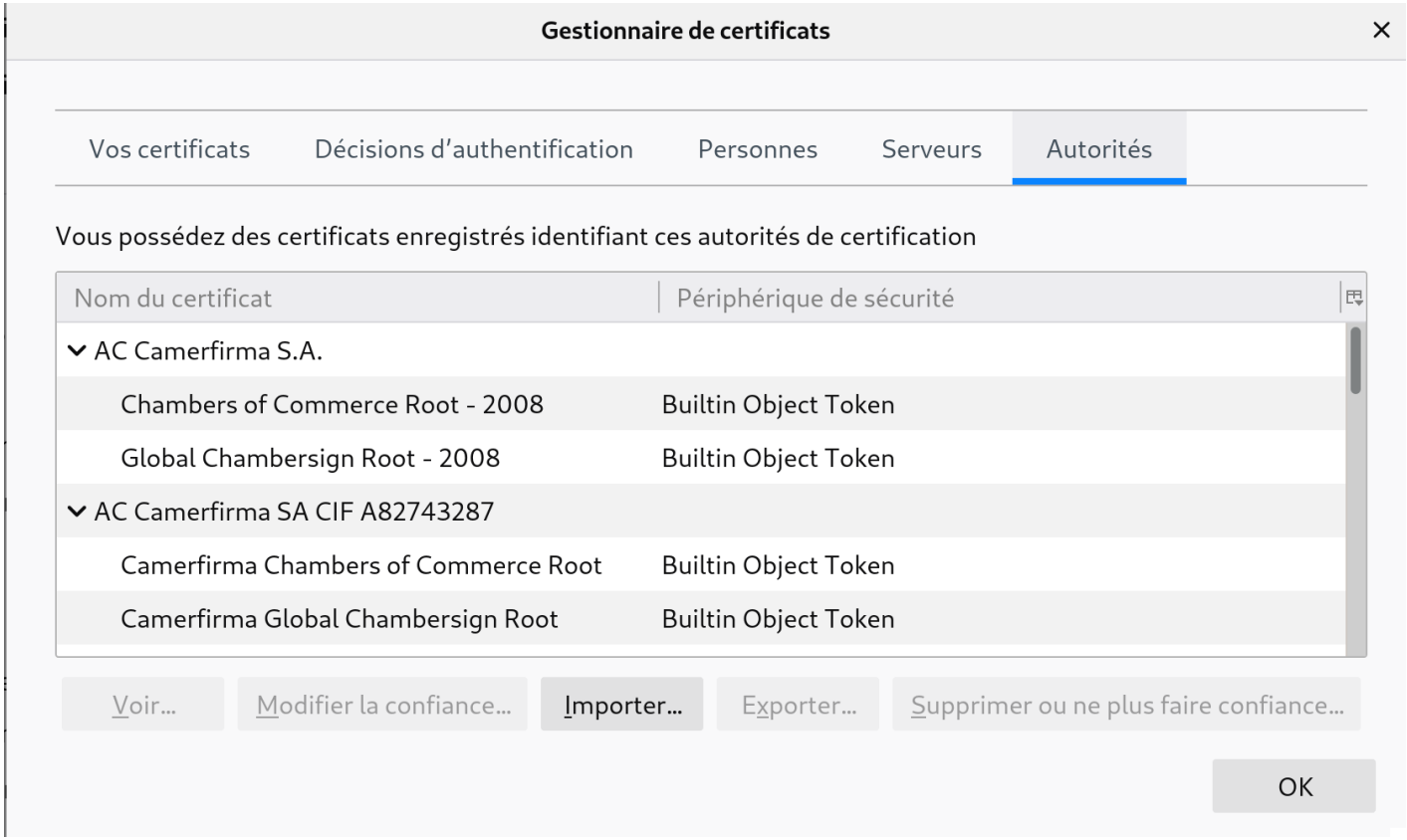


FIGURE – Firefox/préférences/vie privée et sécurité/certificats

En pratique, elle ne signe pas la clé publique entière du site (2048 ou 4096 bits) mais sa somme de contrôle calculée (256 bits) par une fonction de hachage (souvent sha256).

https

Le protocole *https* ajoute une couche *TLS (Transport Layer Security)* au protocole *http* existant.

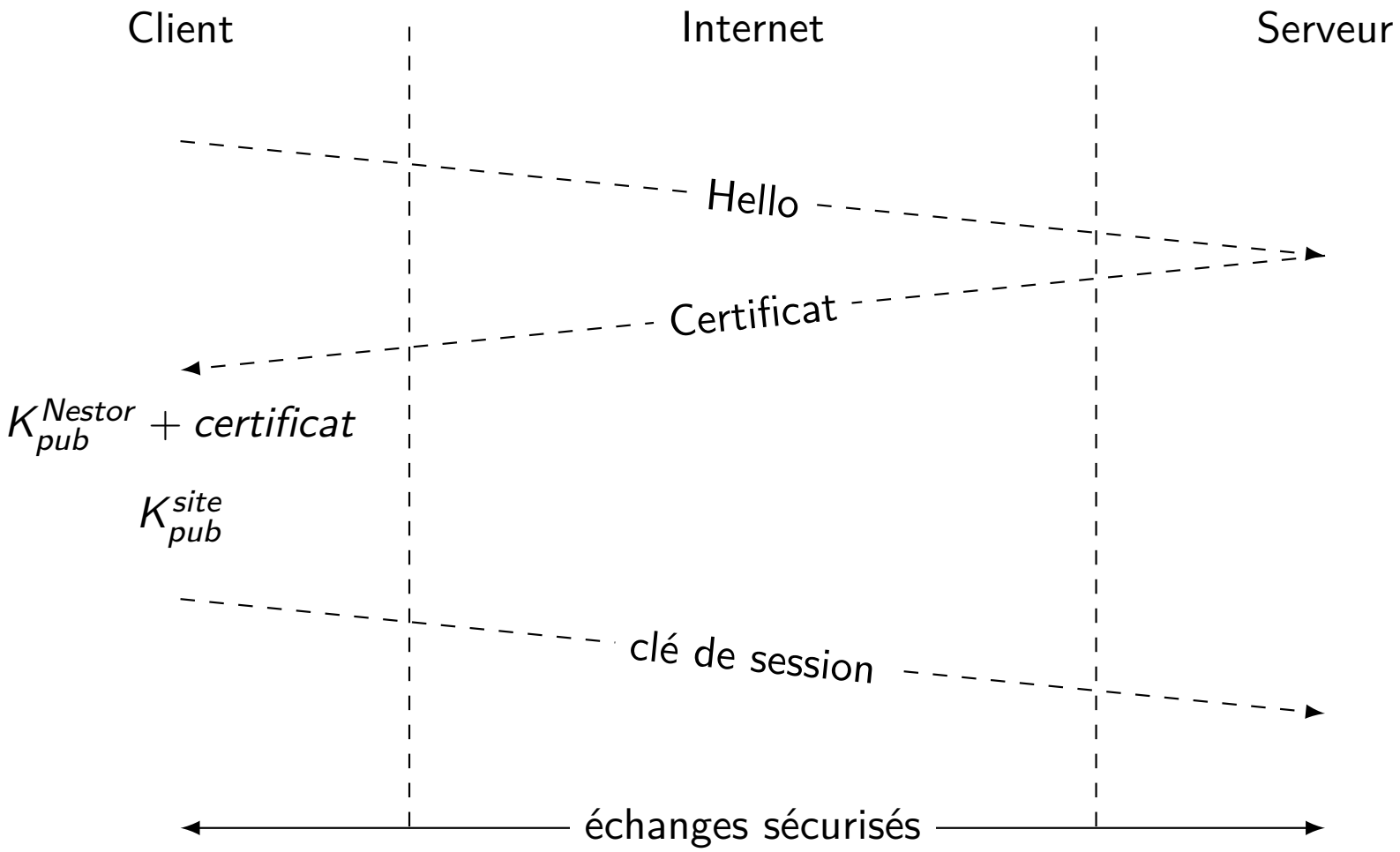


FIGURE – protocole https

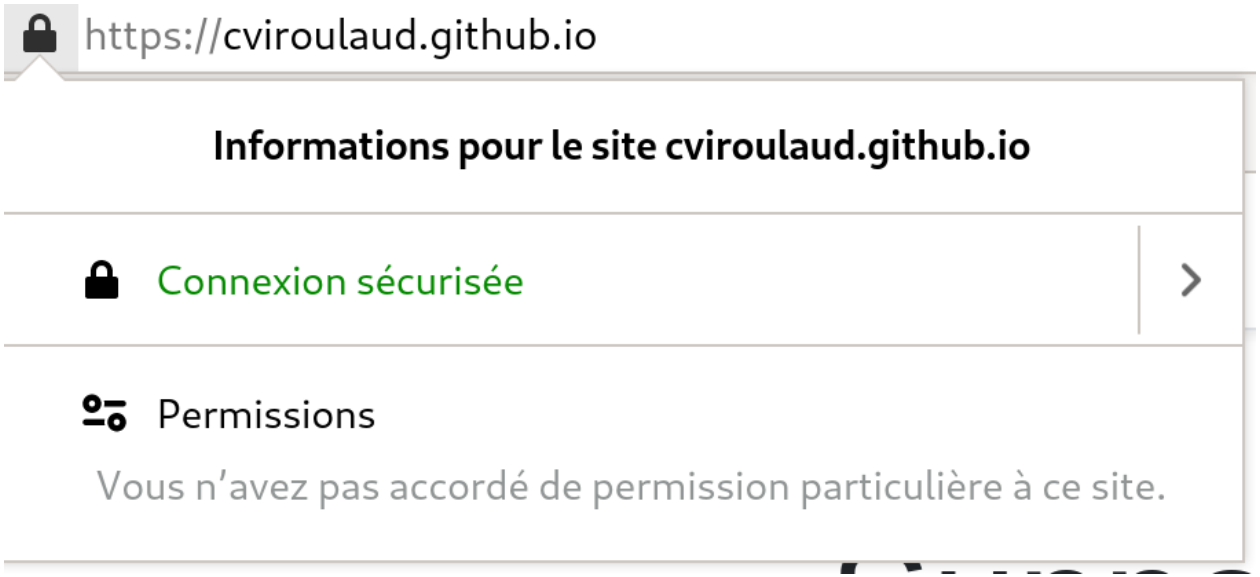


FIGURE – Le cadenas atteste des échanges sécurisés

Détails techniques
Connexion chiffrée (clés TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, 256 bits, TLS 1.2)
La page actuellement affichée a été chiffrée avant d'avoir été envoyée sur Internet.

FIGURE – Exemple

- Protocole TLS (Transport Layer Security)
- ECDHE : Elliptic Curve Diffie-Hellman Ephemeral pour l'échange de clé de session
- RSA pour l'authentification
- Le chiffrement symétrique est assuré par AES128 (Advanced Encryption Standard 128 bits)