Échange de clés

TLS-SEC

Module d’entrée Cryptographie

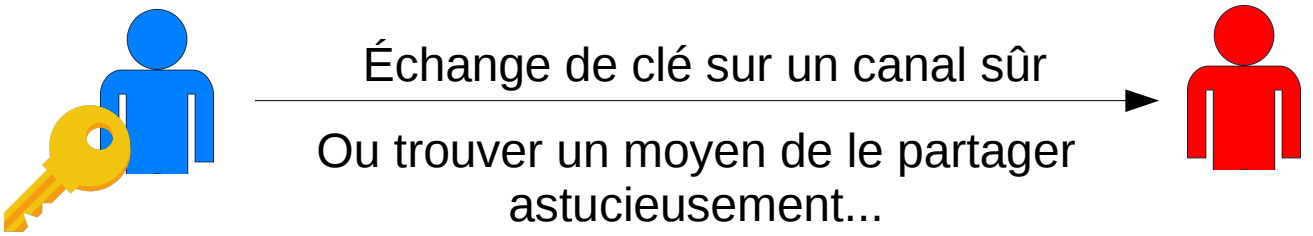
Vincent MIGLIORE

Agenda

- Introduction sur l’échange de clé
- Les premiers échanges de clés
- Les fonctions à trappe
- La dérivation de clés

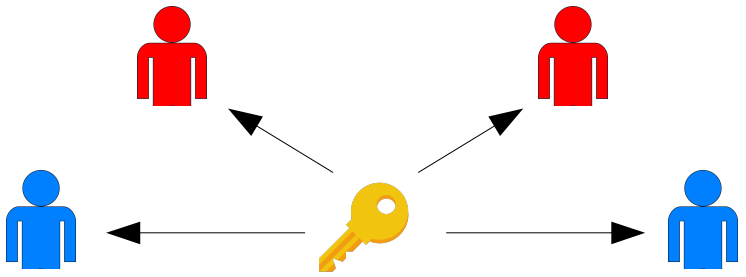
Le partage de clés dans un monde symétrique

- Cas de deux utilisateurs :
  - Si secret déjà partagé, ok. Sinon :



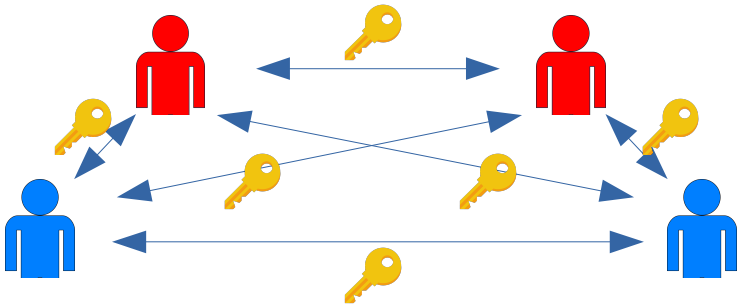
- Cas d'un groupe d'utilisateur :

Partager un secret ?



Dangereux.
Usurpation d'identité, interception de message,...

Secret par couple d'utilisateur ?



Mieux,
mais nécessite un grand nombre de secret : n.(n-1)

Méthode classique : Le tier de confiance

(1)

- A demande à s'enregistrer pour communiquer avec B.
- Ils échangent une clé unique pour communiquer (sécurité + intégrité)

(2)

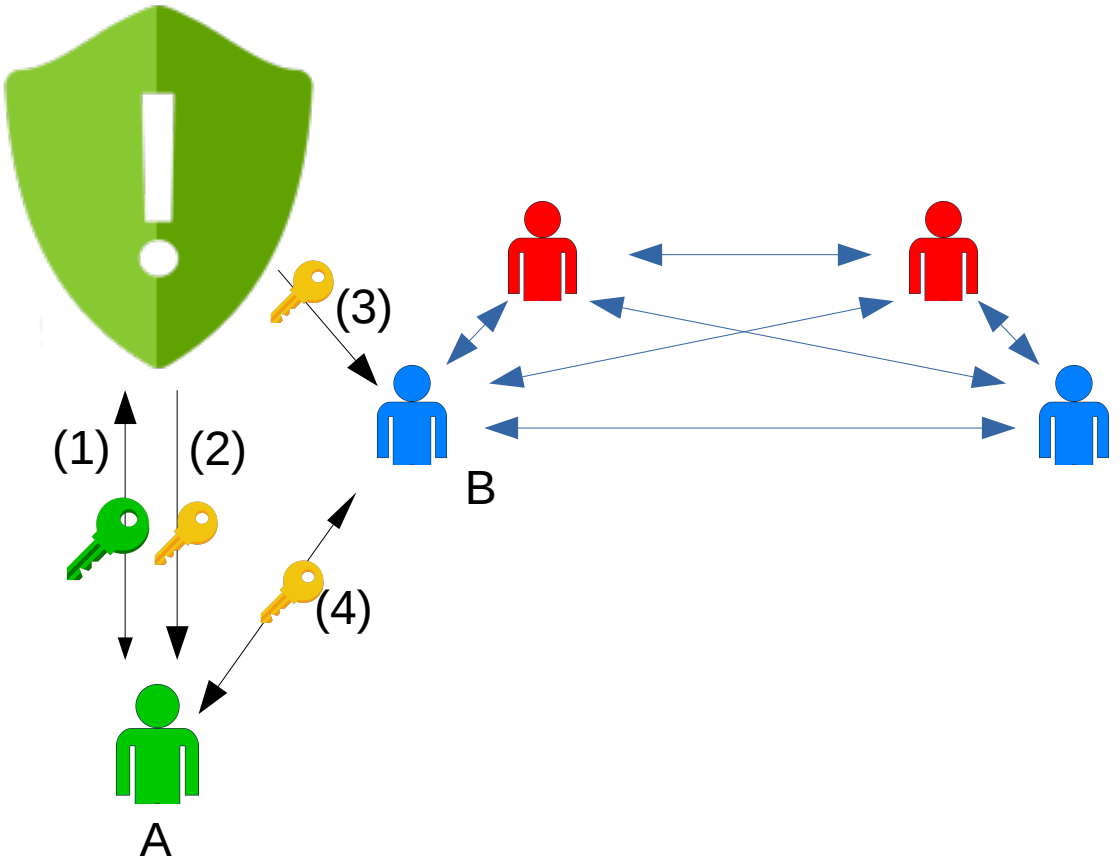
- A fourni au TTP une clef unique pour communiquer avec B

(3)

- Le TTP fourni cette clé à B.

(4)

- A et B peuvent communiquer.



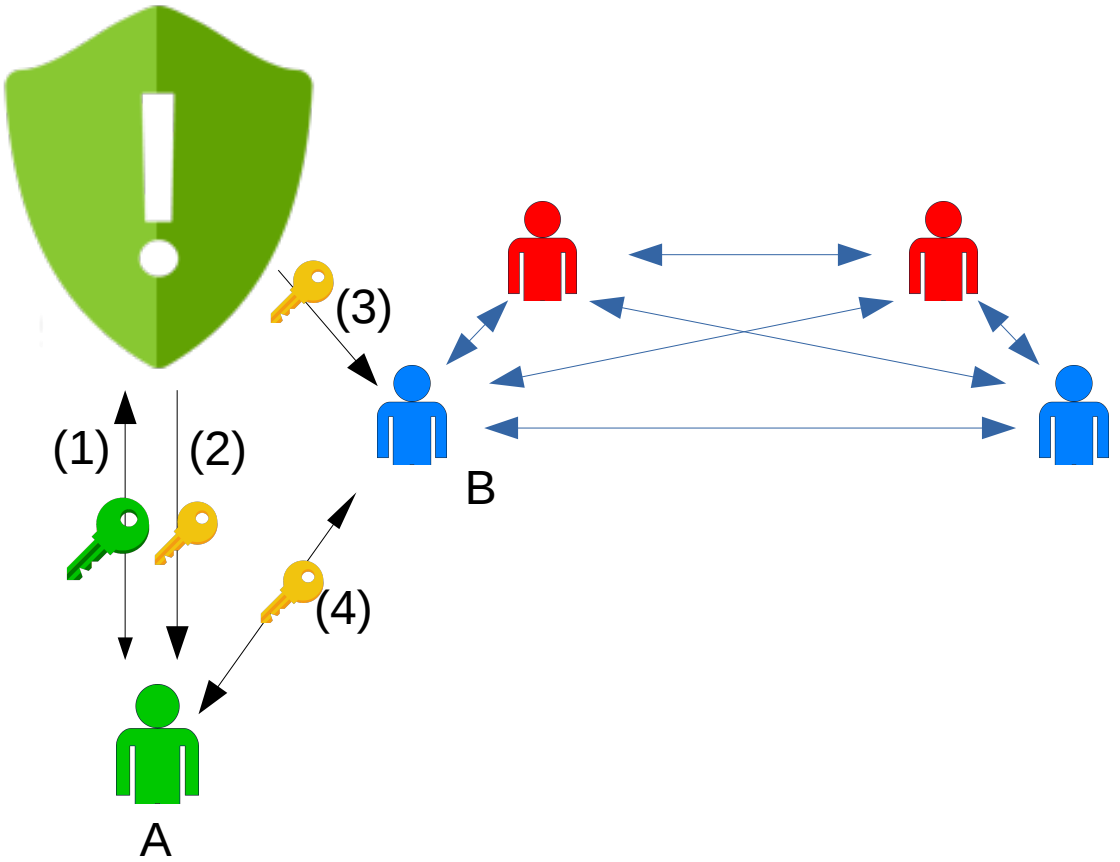
Méthode classique : Le tier de confiance

Remarques :

(1) Le TTP possède toutes les clés → Confiance

(2) Compte tenu que l'on ne peux pas réutiliser systématiquement les mêmes clés, il faut très souvent passer par le tier de confiance.

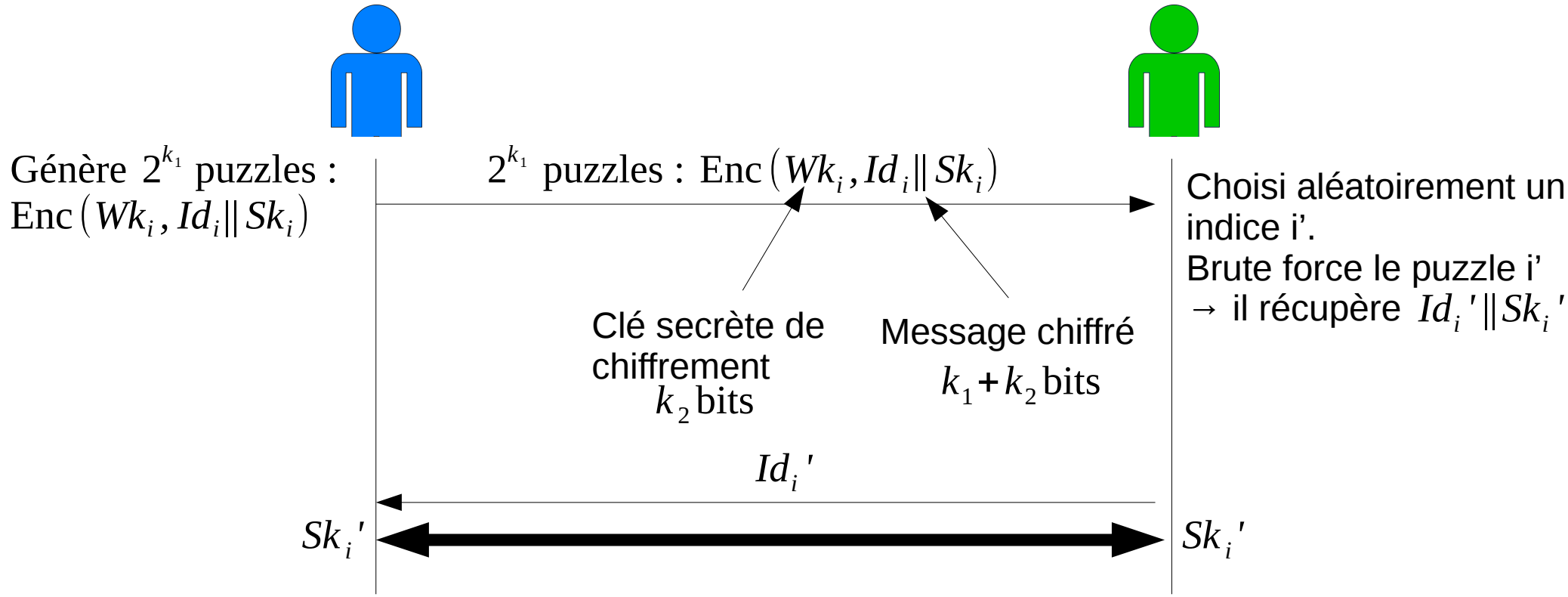
(3) Et si le TTP devient défaillant ? (single point of faillure



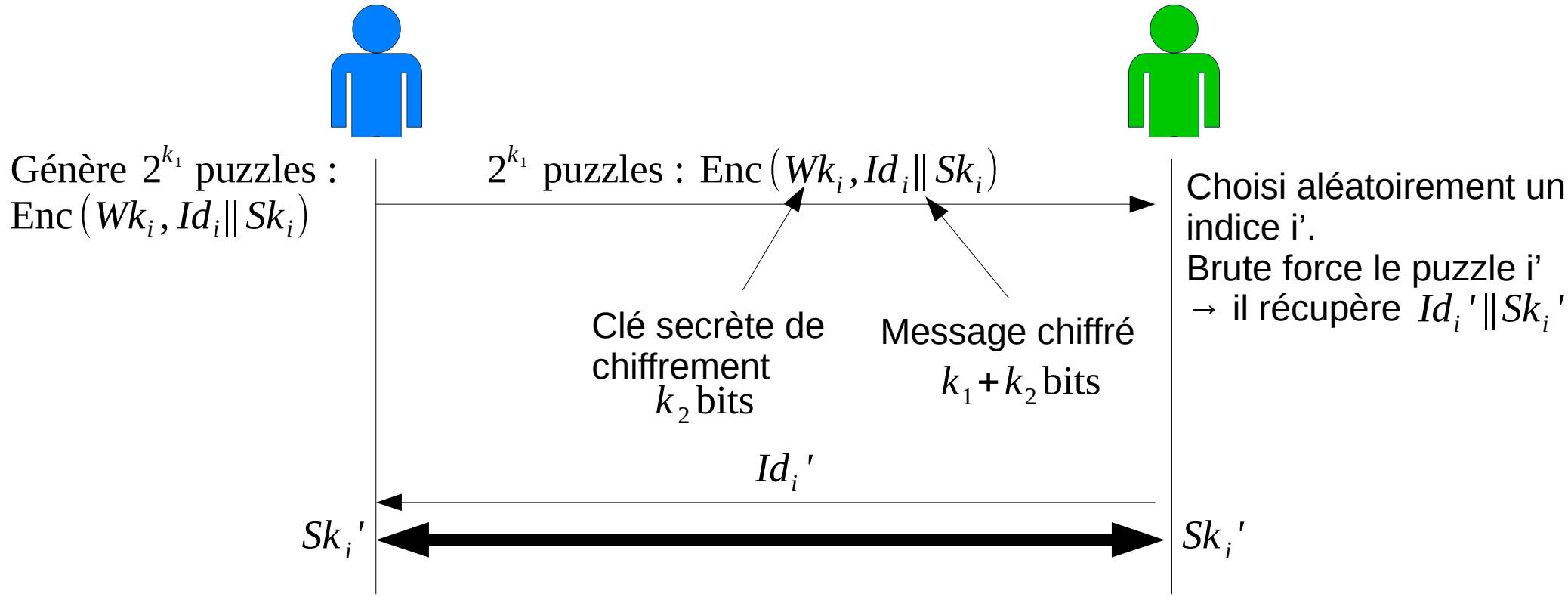
Le problème n’est toujours pas réglé…

Comment on échange le secret ?

Cas historique : Le puzzle de Merkle



Cas historique : Le puzzle de Merkle



- Complexité de brute force pour : $O(2^{k_2})$
- Complexité de brute force pour : $O(2^{k_1+k_2})$

→ Beaucoup trop simple pour l'attaquant...

Diffie-Hellman (1976)

Introduit dans *New directions in cryptography*

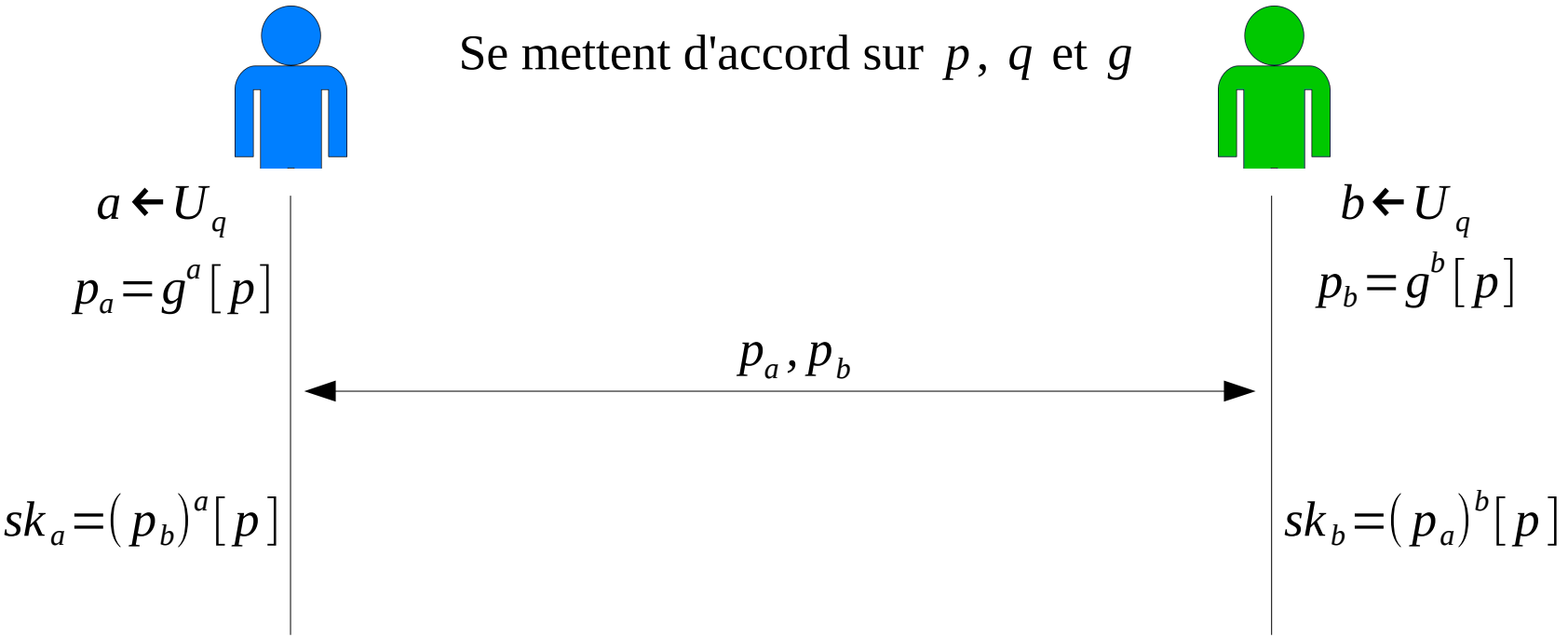
- C’est un mécanisme d’échange de clé.
- Il permet un échange de clé avec un coût :
  - Polynomial pour les deux parties.
  - Exponentiel pour un attaquant.
- Bien que ce mécanisme date de + de 40 ans, il est toujours d’actualité.

Diffie-Hellman (1976)

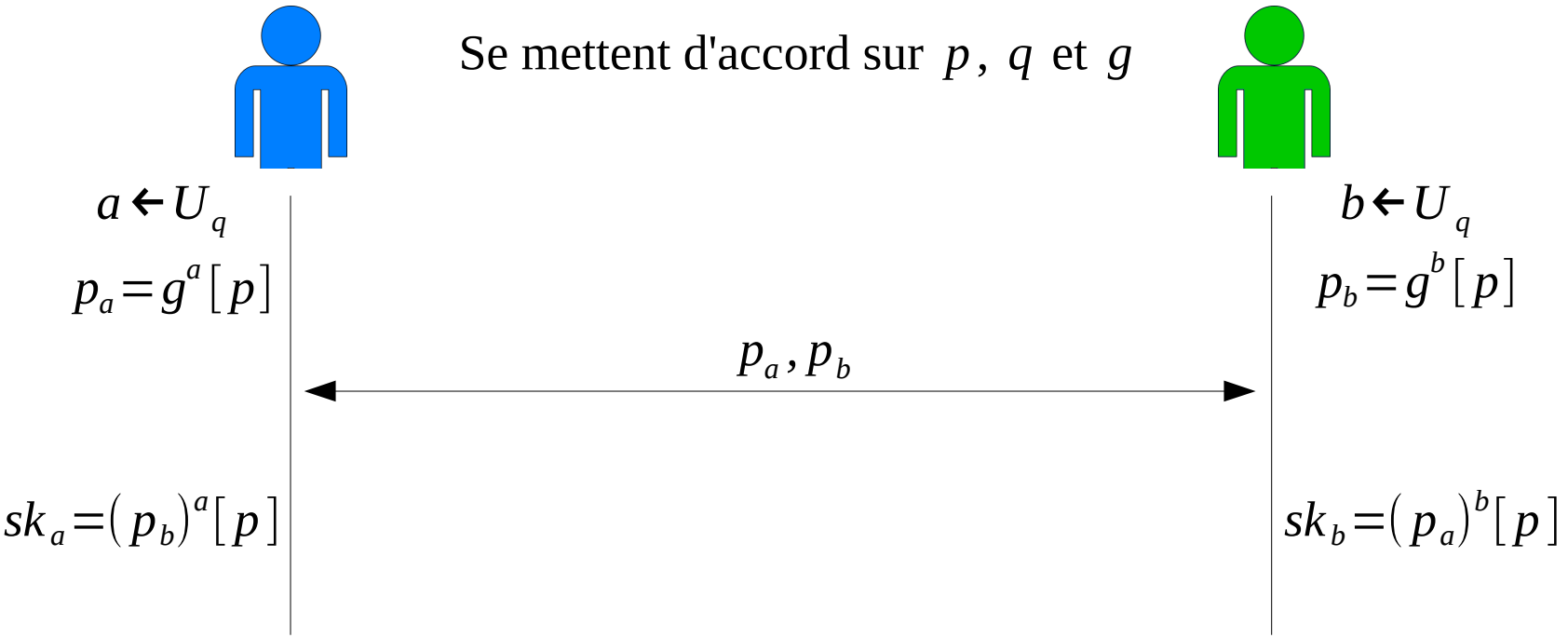
Base mathématique

- La description initiale se place sur un **corps fini** :
  - *p* nombre premier de grande taille (>1024 bits)
  - *q* nombre premier tq *p* = 1 + *K*.*q* (>100 bits)
  - On définit Z_p comme l'ensemble des entiers modulo *p* (i.e 0, 1, …, *p*-1)
  - On peut trouver un entier *g* (appelé générateur), tq :
    - $\forall (i,j) \in Z_q, i \neq j \rightarrow g^i \ [p] \neq g^j \ [p]$
    - $g^q \equiv g \ [p]$

Diffie-Hellman (1976)

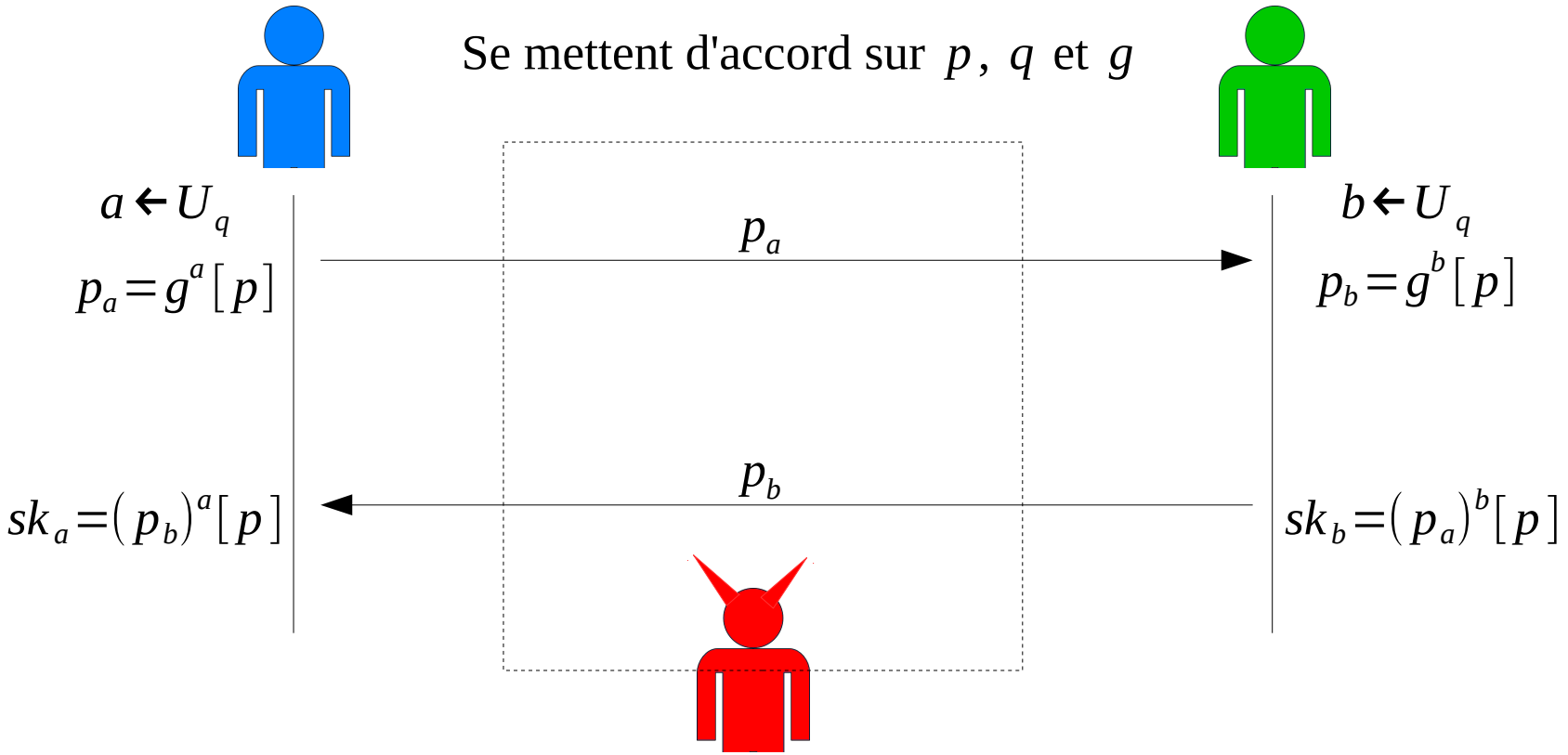


Diffie-Hellman (1976)



- Question :
  - Montrez que $sk_a = sk_b$
  - Le temps de calcul est-il polynomial ?

« Man in the Middle » et Diffie-Hellman (1976)

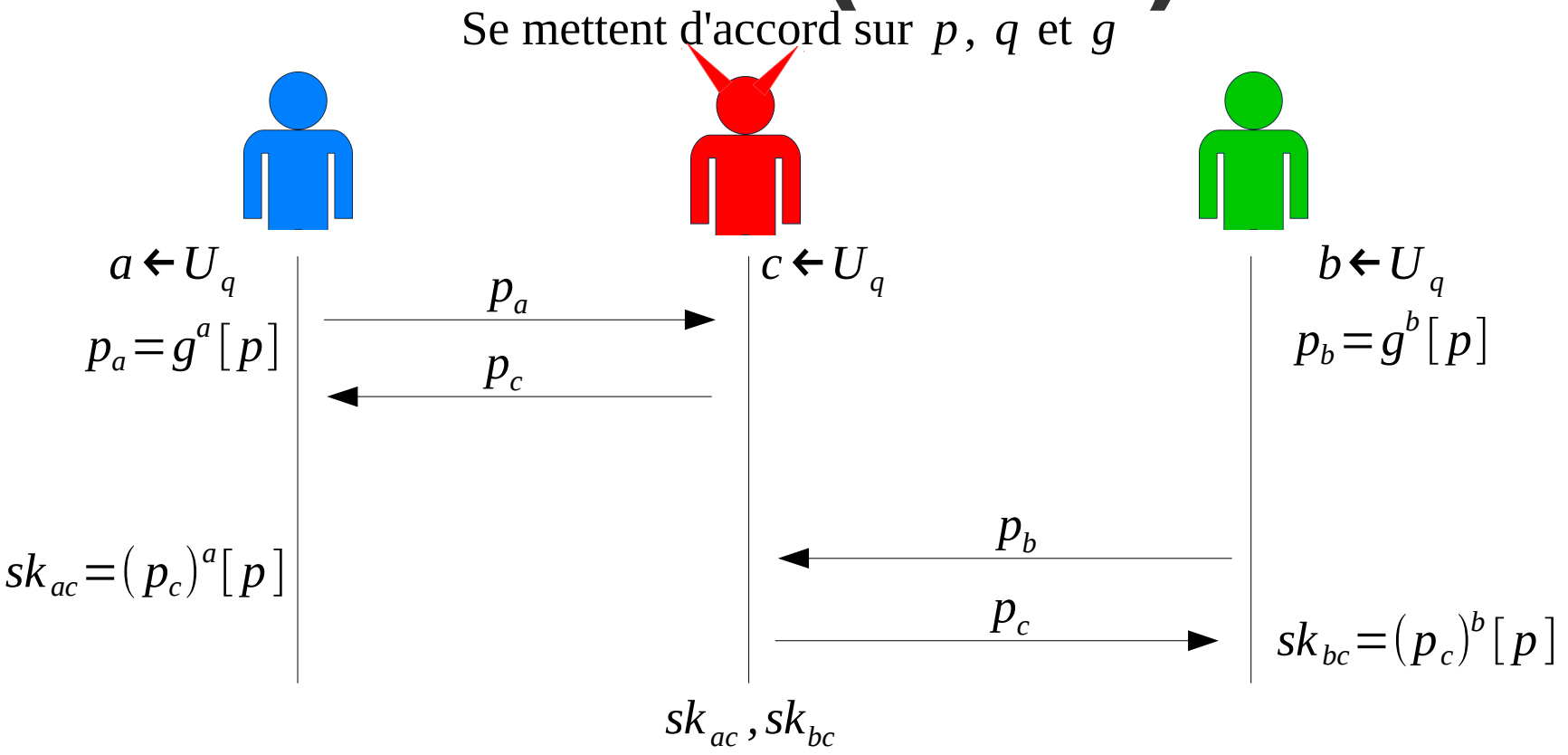


Si l'attaquant ne fait qu'écouter, il ne peux pas récupérer le secret partagé (Attaquant passif).

Question :

A votre avis, que peut faire un attaquant qui modifie les communications ? (attaquant actif)

« Man in the Middle » et Diffie-Hellman (1976)



L'attaquant actif est capable de se placer au milieu de la communication et intercepter toutes les communications → « MITM Attack »

Solutions

Simple : obtention des clés par un canal sûr et mémorisation (coût linéaire)
Sans canal sûr : il faudrait un moyen de certifier les clés

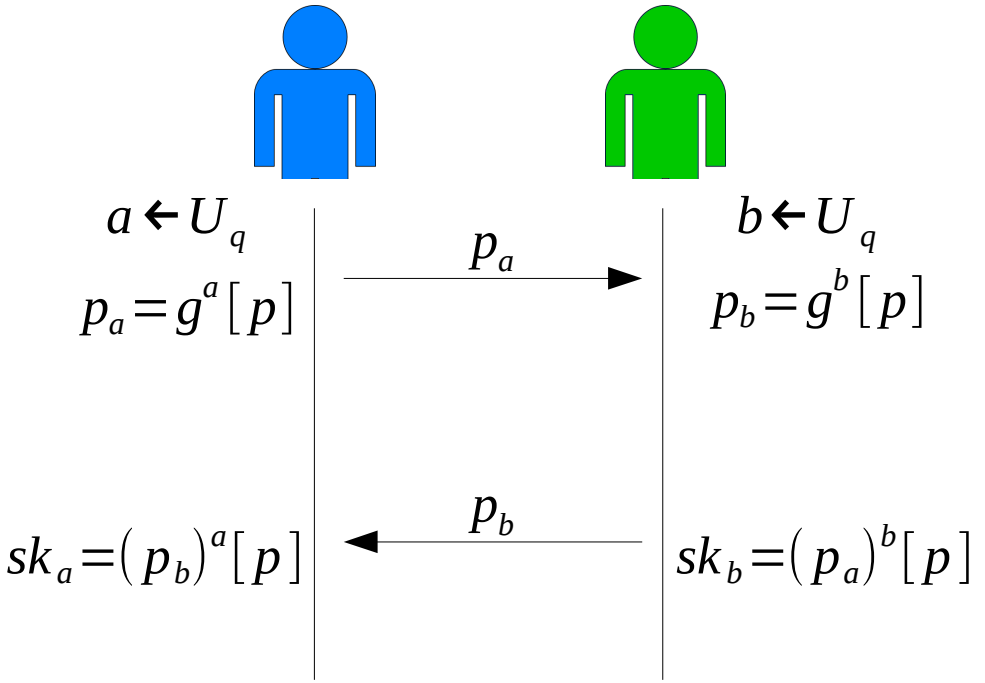
Autres attaques sur Diffie-Hellman (1976)

Et si…

Quelqu'un volait la clé secrète de A sans que A s'en rende compte …
Les échanges de clés futurs avec A seraient sécurisés ?

Et si…

Quelqu'un avait enregistré tous les échanges passés et les communications chiffrées, puis avait volé la clé secrète de A. Pourrait-il déchiffrer les échanges précédents ?



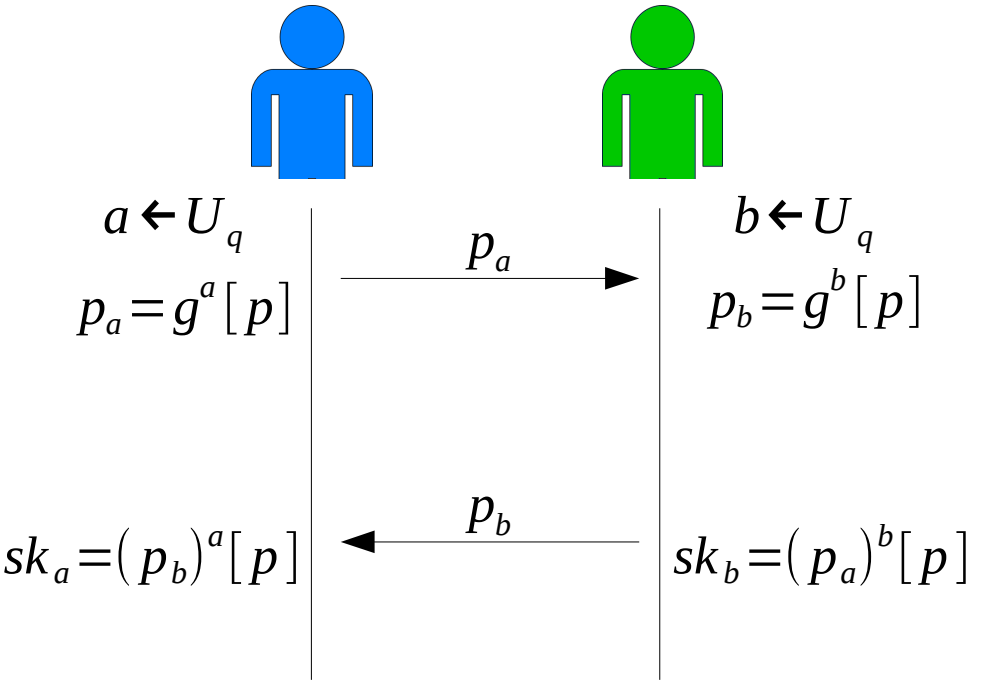
Autres attaques sur Diffie-Hellman (1976)

Et si…

Quelqu'un volait la clé secrète de A sans que A s'en rende compte …
Les échanges de clés futurs avec A seraient sécurisés ?
Non

Et si…

Quelqu'un avait enregistré tous les échanges passés et les communications chiffrées, puis avait volé la clé secrète de A. Pourrait-il déchiffrer les échanges précédents ?



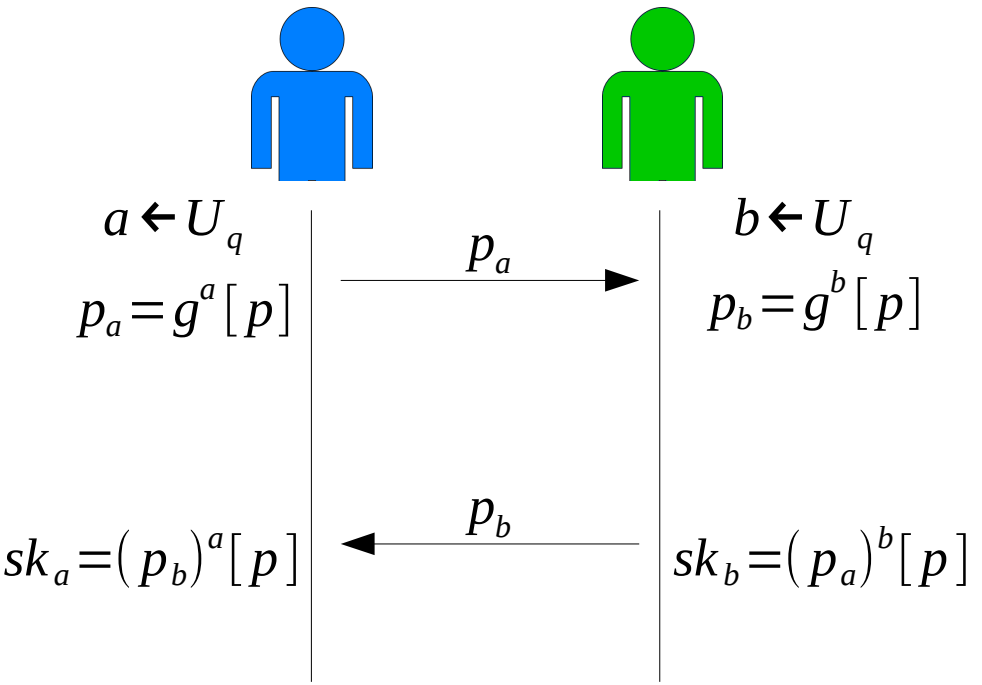
Autres attaques sur Diffie-Hellman (1976)

Et si…

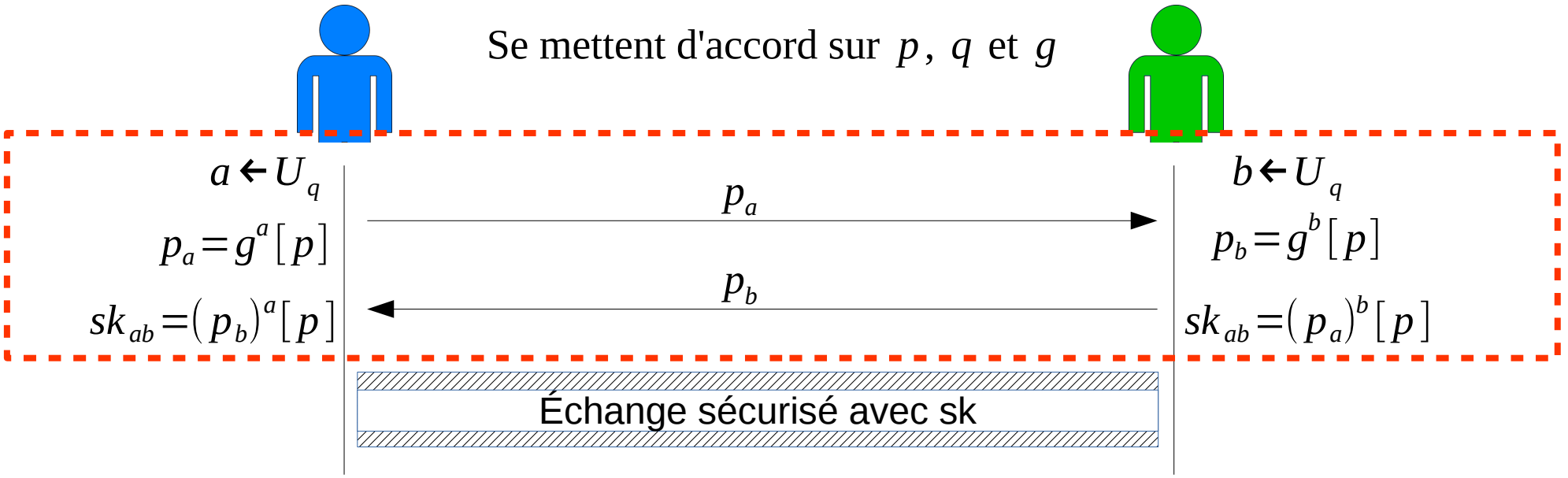
Quelqu'un volait la clé secrète de A sans que A s'en rende compte …
Les échanges de clés futurs avec A seraient sécurisés ?
Non

Et si…

Quelqu'un avait enregistré tous les échanges passés et les communications chiffrées, puis avait volé la clé secrète de A. Pourrait-il déchiffrer les échanges précédents ?
Oui
(pas de sécurité persistante)



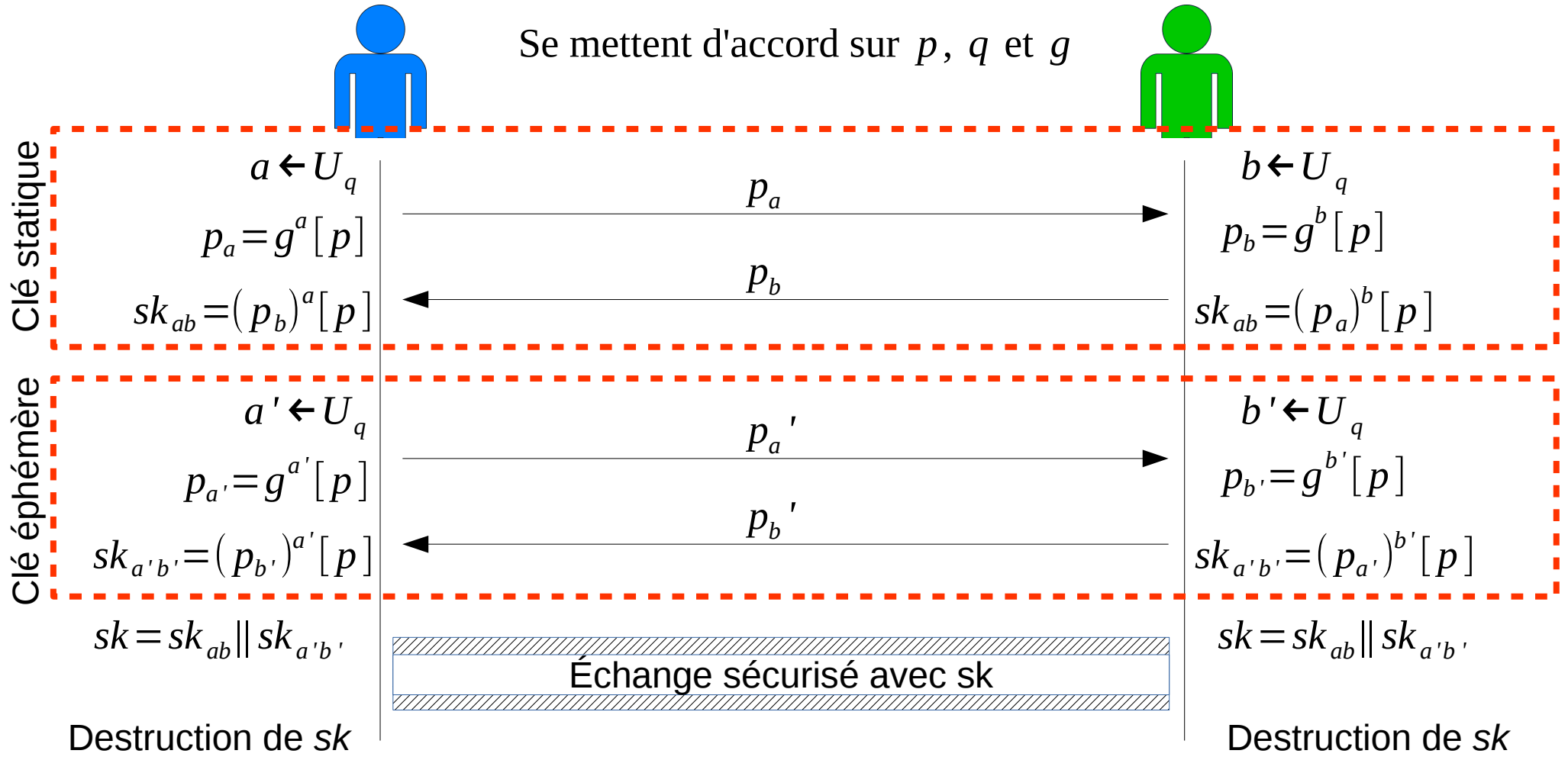
Sécurité Persistante avec Diffie-Hellman



Question :

- A votre avis, quelle méthode peut-on utiliser pour obtenir une sécurité persistante ?

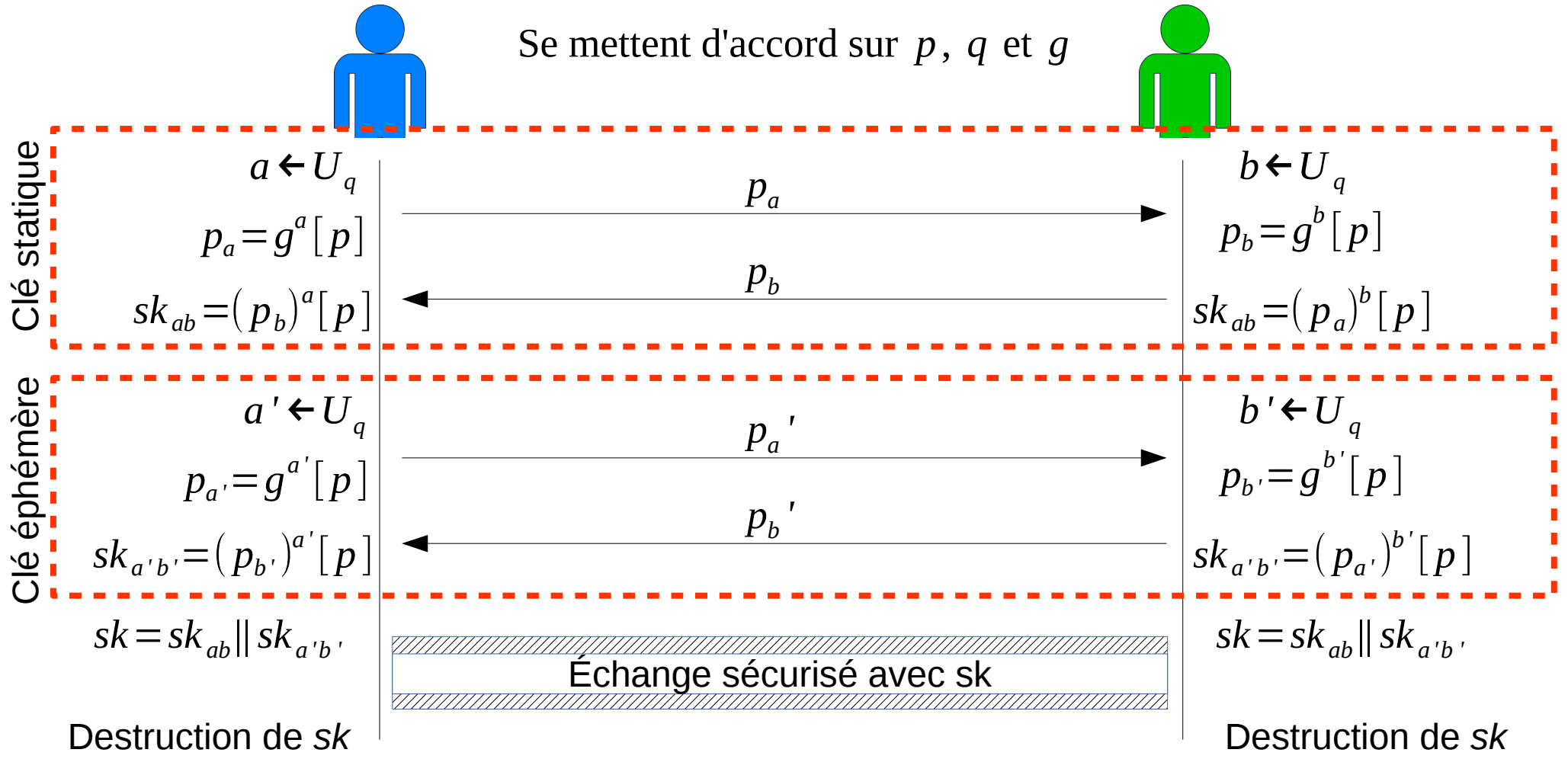
Sécurité Persistante avec Diffie-Hellman



Question :

- Que se passe-t-il si un attaquant arrive à prendre le contrôle de votre serveur ?

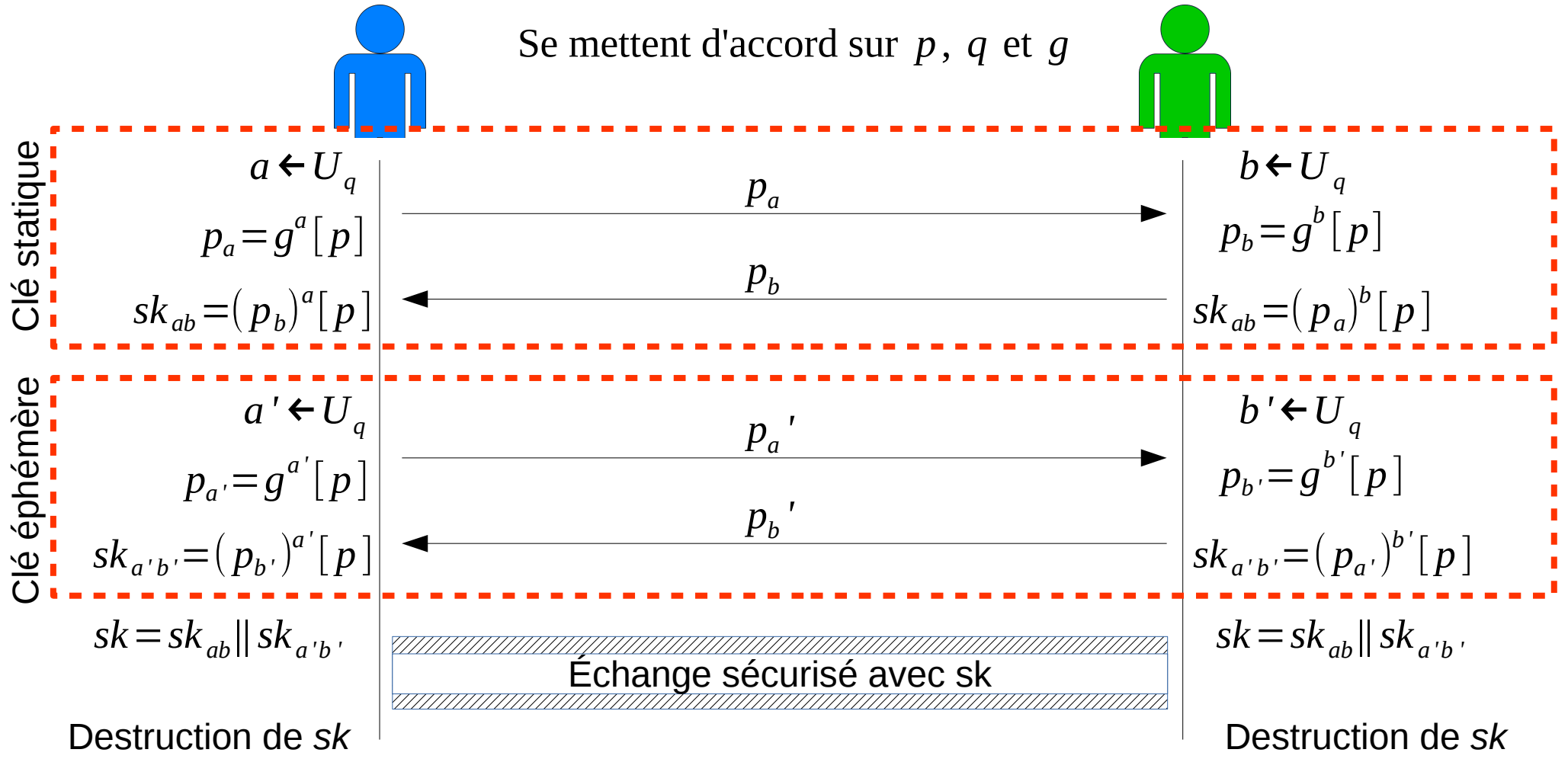
Sécurité Persistante avec Diffie-Hellman



Question :

- Que se passe-t-il si un attaquant arrive à prendre le contrôle de votre serveur ?
- Il récupère sk_{ab}, mais ne peux pas déchiffrer votre échange → Sécurité Persistante

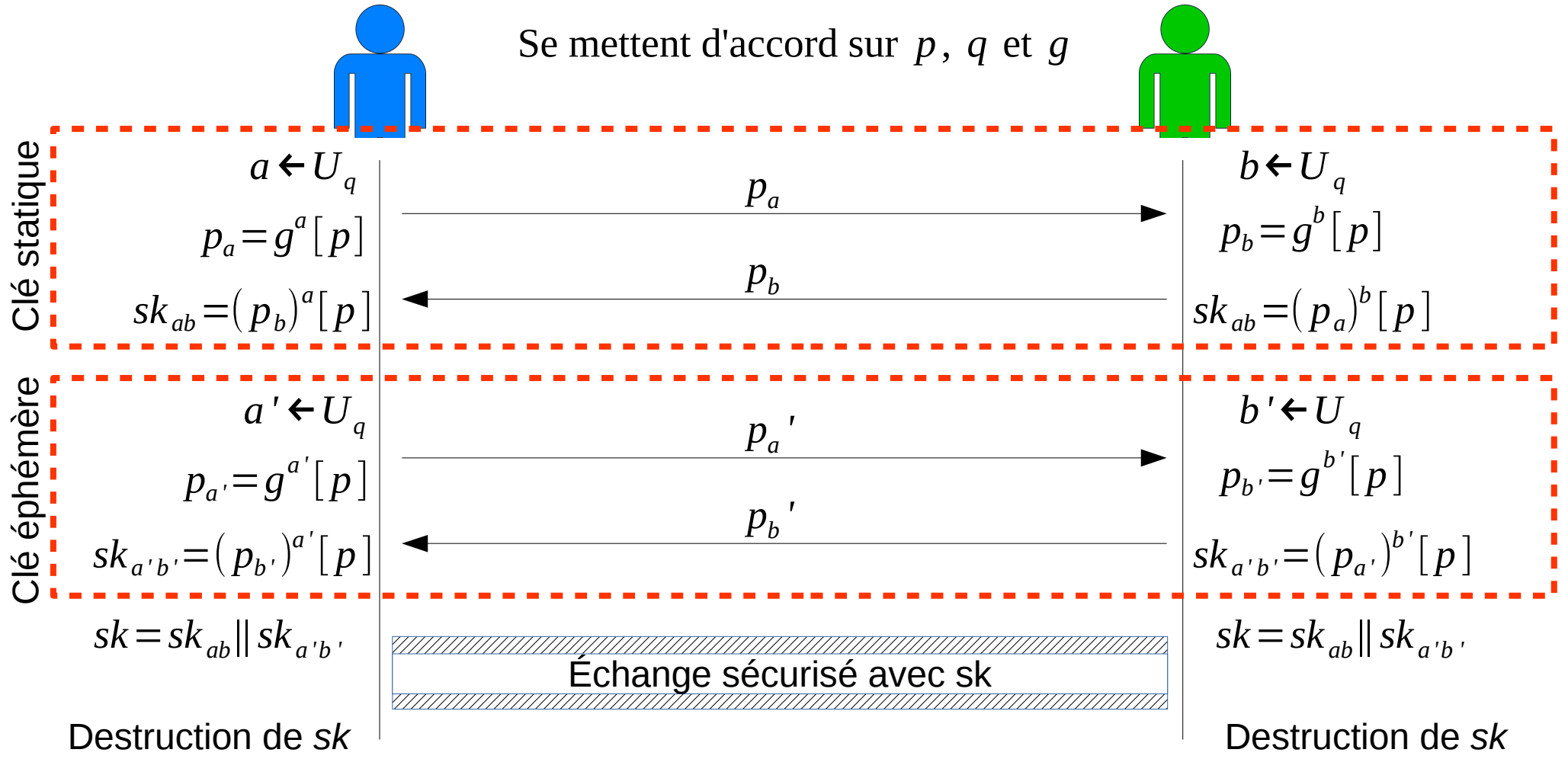
Sécurité Persistante avec Diffie-Hellman



Question :

- Quels échangent demandent une authentification pour éviter une attaque « MITM » ?

Sécurité Persistante avec Diffie-Hellman



Question :

- Quels échangent demandent une authentification pour éviter une attaque « MITM » ?
- Le premier échange de clé seulement, les autres ne sont pas obligatoires.

Que propose le Standard associé à l'échange de clé Diffie-Hellman (1976) ?

NIST SP 800 56A

- Différentes classes selon le nombre de secrets statiques/éphémères :
  - C(2e,2s) : 2 éphémères, 2 statiques
  - C(2e,0s) : 2 éphémères, 0 statiques
  - C(1e,2s) : 1 éphémère , 2 statiques
  - C(1e,1s) : 1 éphémère , 1 statique
  - C(0e,2s) : 0 éphémère , 2 statiques

Que propose le Standard associé à l'échange de clé Diffie-Hellman (1976) ?

NIST SP 800 56A

- Différentes classes selon le nombre de secrets statiques/éphémères :
  - C(2e,2s) : Identification + sécurité persistante
  - C(2e,0s) : Identification par un autre mécanisme requis
  - C(1e,2s) : Identification + variation clé éphémère unilatérale
  - C(1e,1s) : Identification unilatérale + variation clé éphémère unilatérale
  - C(0e,2s) : Identification + absence de sécurité persistante
- Vous pouvez retrouver des justifications page 112 du standard rev3.

Sécurité de Diffie-Hellman (1976)

- Supposition initiale :
  Complexité exponentielle pour l'attaquant.
- La sécurité de Diffie-Hellman repose sur le problème difficile suivant :
  - Connaissant (g, p, g^a, g^b) trouver $x = g^{ab}$ $[p]$

    (Computational DH : CDH)
  - CDH est réputé difficile à résoudre.

Sécurité de Diffie-Hellman (1976)

- L'approche pour évaluer la sécurité de DH est de passer par le logarithme discret :
  - Connaissant y trouver x tel que $y = g^x \ [p]$
- Ainsi, Connaissant (g, p, g^a, g^b) on applique le logarithme discret pour trouver *a* et *b*, puis on en déduit g^{ab}

Relation entre Diffie-Hellman et le Logarithme discret

- Casser le Logarithme Discret → Casser Diffie-Hellman : ok
- Casser Diffie-Hellman → casser Logarithme Discret : Pas de preuve depuis sa construction.
- Il n’existe pas de preuve non plus que le probleme CDH soit plus simple que le logarithme discret.

Complexité du logarithme discret

Notation L de la complexité

- $L_p[\alpha, c] = 2^{(c+o(1))(\log p)^{\alpha}(\log\log p)^{(1-\alpha)}}$
- Permet d'étaler la complexité d'un algorithme entre polynomial et exponentiel :
  - $\alpha = 0 \rightarrow L_p[0, c] = 2^{(c+o(1))(\log\log p)} = (\log p)^{(c+o(1))}$
  - $\alpha = 1 \rightarrow L_p[1, c] = 2^{(c+o(1))(\log p)}$
- Meilleur algorithme aujourd'hui : crible algébrique :
  - $L_p[1/3, 1.92] = 2^{(1.92+o(1))(\log p)^{1/3}(\log\log p)^{2/3}}$
- Question :
  - On veut doubler la complexité (i.e. doubler la valeur dans l'exposant), quel impact sur la taille de *p* ?

Complexité du logarithme discret

Notation L de la complexité

- $L_p[\alpha, c] = 2^{(c+o(1))(\log p)^{\alpha}(\log\log p)^{(1-\alpha)}}$
- Permet d'étaler la complexité d'un algorithme entre polynomial et exponentiel :
  - $\alpha = 0 \rightarrow L_p[0, c] = 2^{(c+o(1))(\log\log p)} = (\log p)^{(c+o(1))}$
  - $\alpha = 1 \rightarrow L_p[1, c] = 2^{(c+o(1))(\log p)}$
- Meilleur algorithme aujourd'hui : crible algébrique :
  - $L_p[1/3, 1.92] = 2^{(1.92+o(1))(\log p)^{1/3}(\log\log p)^{2/3}}$
- Question :
  - On veut doubler la complexité (i.e. doubler la valeur dans l'exposant), quel impact sur la taille de *p* ?
  - Cela revient à multiplier log *p* par $2^3 = 8$

Logarithme discret

Recommandations basées sur le crible algébrique pour Log discret sur les corps finis

Niveau de sécurité	log *p*	log *q*
80 bits	1024	160
112 bits	2048	224
112 bits	2048	256
> 112 bits	Possible, mais corps finis non recommandés	

Extrait de la norme RFC 5114

Utilisation du secret commun DH comme clé symétrique

A votre avis, peut-on utiliser directement le secret commun obtenu à la suite d'un échange DH pour un chiffrement symétrique ?

Utilisation du secret commun DH comme clé symétrique

A votre avis, peut-on utiliser directement le secret commun obtenu à la suite d'un échange DH pour un chiffrement symétrique ?

NON :

- L'hypothèse de sécurité de DH garanti seulement que (g^x,g^y,g^{xy}) n'est pas distinguable de (g^x,g^y,g^z).
- Hors, (g^x,g^y,g^{xy}) est en général distinguable de (g^x,g^y,R), avec R chaîne aléatoire.
- Raison simple, il n'existe pas de correspondance $R \leftrightarrow g^{xy}$ pour tout R.

Utilisation du secret commun DH comme clé symétrique

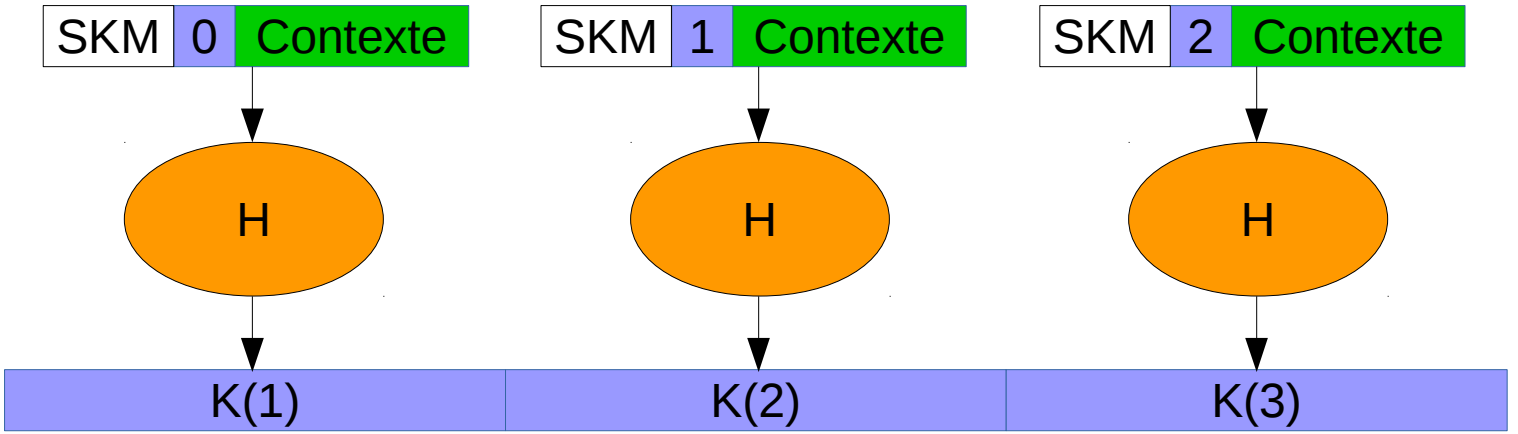
A votre avis, peut-on utiliser directement le secret commun obtenu à la suite d'un échange DH pour un chiffrement symétrique ?

- Solution standard :

  Utiliser une fonction de dérivation de clé (KDF).

- Une KDF permet :
  - De transformer une source d'aléa avec biais vers une source d'aléa uniforme.
  - A dériver une clé différente par contexte d'utilisation.

Cas 1 : Dérivation de clé en 1 étape

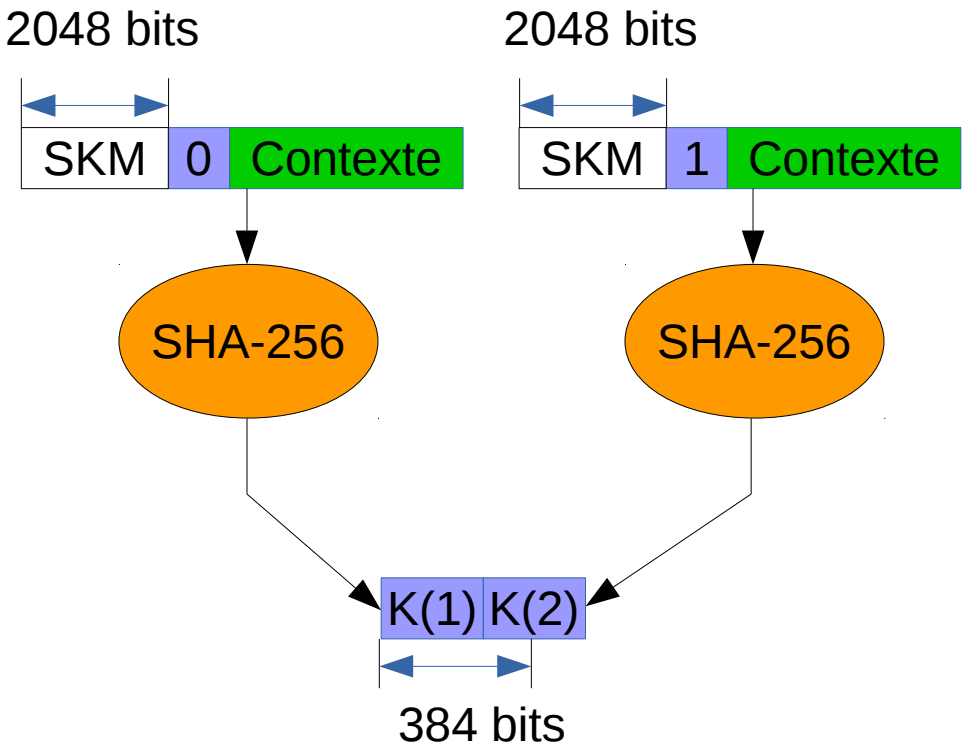


SKM = Secret Key Material = Secret partagé

Documents de référence :

- ISO 18033-2
- IEEE P1363A
- PKCS#1 v2
- NIST SP 800 56C

Cas 1 : Dérivation de clé en 1 étape

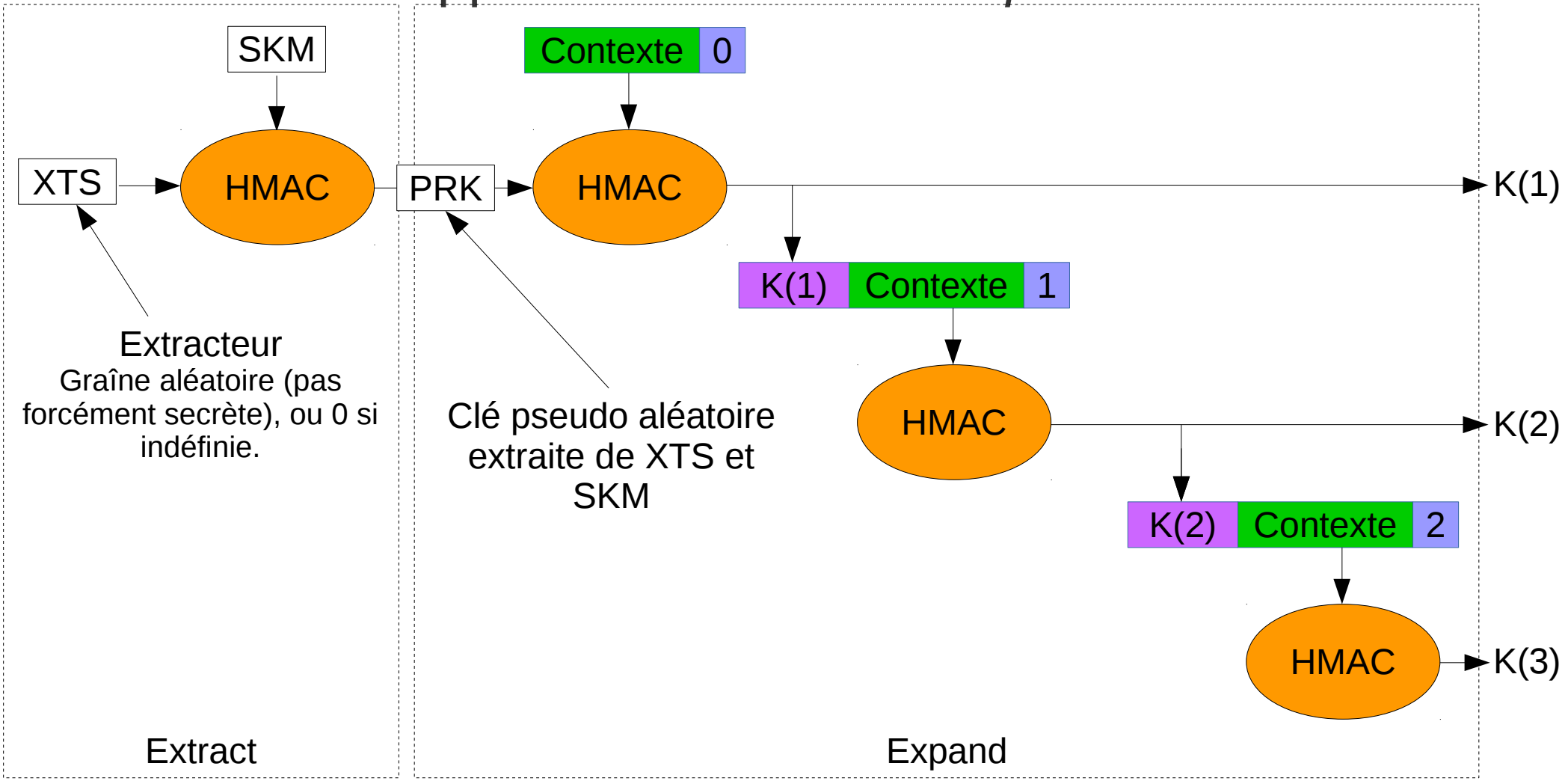


Exemple :

- Utilisation de la fonction de hachage SHA-256
- Paramètres Diffie-Hellman : log p = 2048 bits, log q = 256 bits
- Cible : TLS, demandant 48 octets en sortie de la dérivation de clé

Cas 2 : Dérivation de clé en 2 étapes

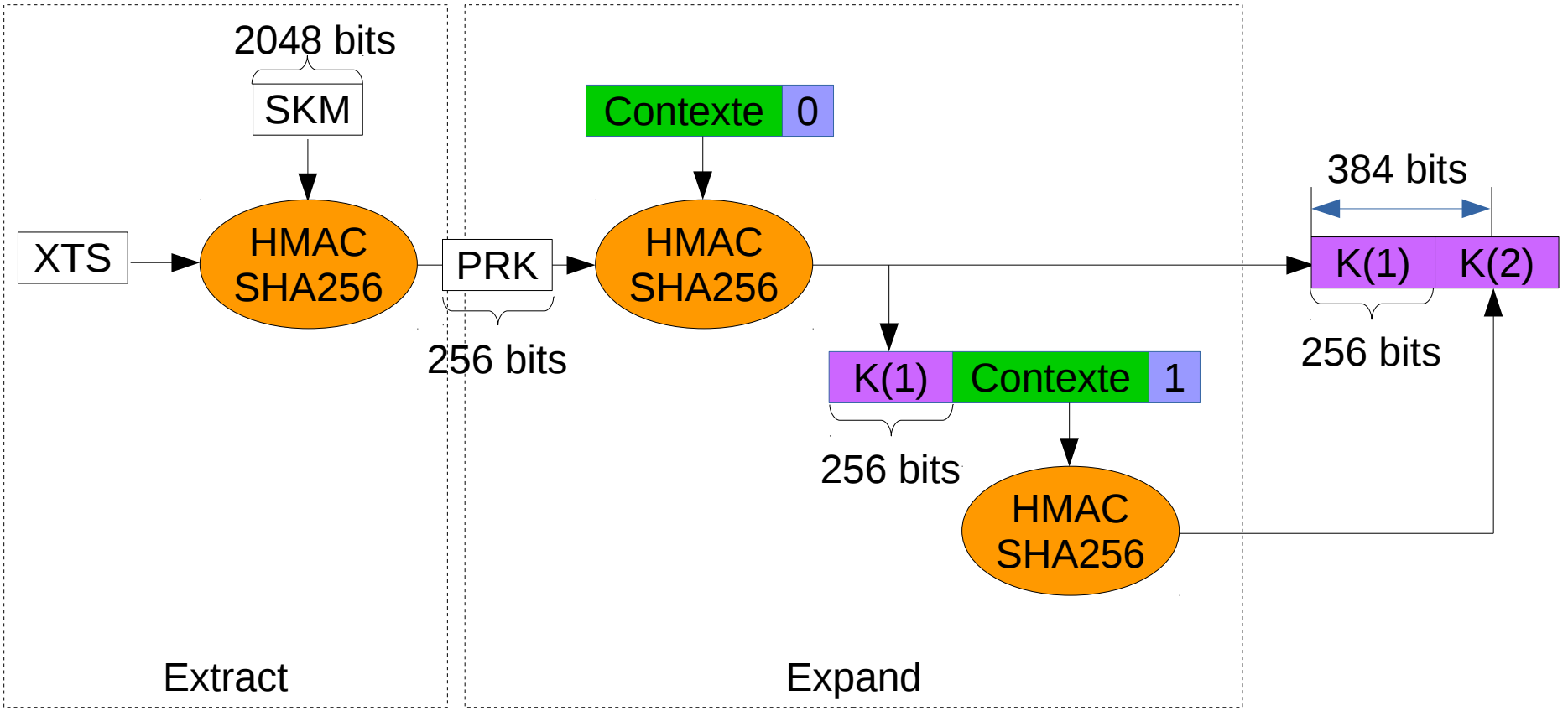
Méthode appelée *Extract-then-Expand*



Cas 2 : Dérivation de clé en 2 étapes

Exemple :

- Utilisation de la fonction de hachage SHA-256
- Paramètres Diffie-Hellman : log p = 2048 bits, log q = 256 bits
- Cible : TLS, demandant 48 octets en sortie de la dérivation de clé



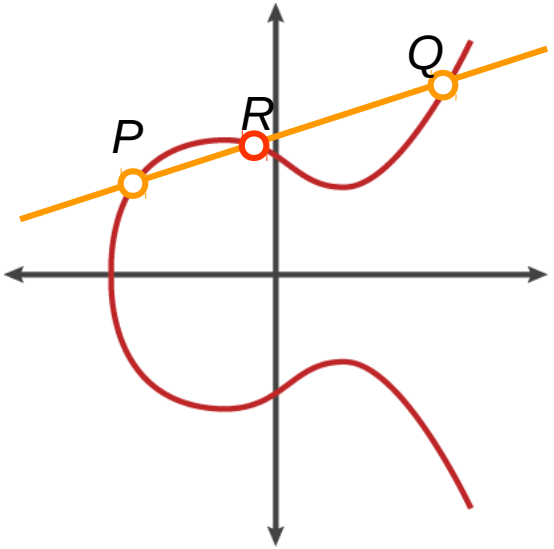
Améliorer la sécurité/performance de Diffie-Hellman

Diffie-Hellman dans un groupe générique

- Diffie-Hellman n'est pas restreint aux corps finis. Construction :
  - Choisir un groupe G muni d'une opération ○
  - Choisir *g* générant un sous-groupe G' de G de taille *q*.
  - Connaissant $\underbrace{g \circ g \circ \ldots \circ g}_{x \text{ fois}}$, trouver *x* est difficile.

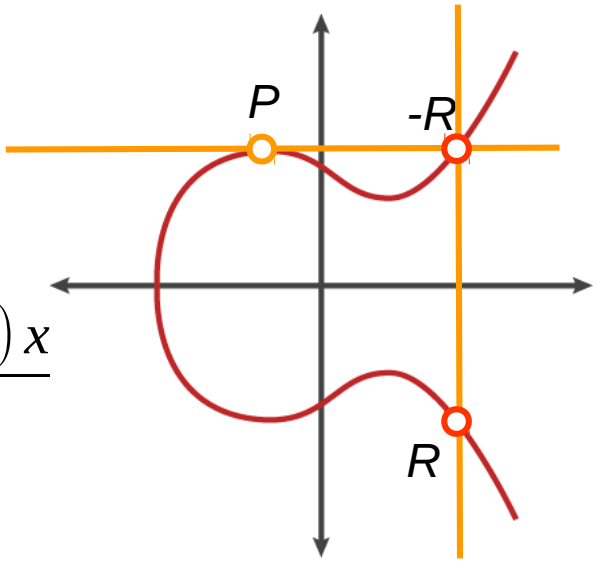
Diffie-Hellman sur les courbes elliptiques

- **Cas classique et standardisé : DH sur les courbes elliptiques**
  - $y^2 = x^3 + a.x + b$
  - Addition de points : $P + Q = R$
    - Intersection courbe et droite PQ
    - Droite : $y = x\,s + t\,, s = \frac{y_Q - y_P}{x_Q - x_P}, t = \frac{x_P\,y_Q - x_Q\,y_P}{x_Q - x_P}$
    - $y_R = s^2 - y_P - y_Q$
    - $x_R = s(y_P - y_R) - y_P$



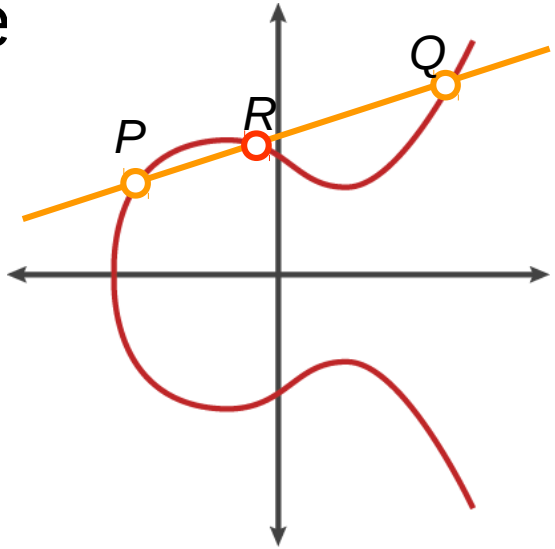
Diffie-Hellman sur les courbes elliptiques

- **Cas classique et standardisé : DH sur les courbes elliptiques**
  - $y^2 = x^3 + a.x + b$
  - Doublement de points : $P + P = R$
    - Intersection courbe et tangente
    - Droite : $y = x\,s + t\,, s = \frac{3\,x_P^2 + a}{2\,y_P}\,, t = y_P - \frac{(3\,x_P^2 + a)\,x}{2\,y_P}$
    - $y_R = s^2 - 2\,y_P$
    - $x_R = s(y_P - y_R) - y_P$



Diffie-Hellman sur les courbes elliptiques

- **Algorithme optimisé pour l'addition : [Cohen-Miyaji-Ono 98]**
  - Utilisation de coordonnées projective
  - $(x,y) \leftrightarrow (X,Y,Z)$, $x = X/Z$, $y = Y/Z$




```
Input: P = (X1,Y1,Z1), Q = (X2,Y2,Z2)
Y1Z2 = Y1*Z2; X1Z2 = X1*Z2; Z1Z2 = Z1*Z2
u = Y2*Z1-Y1Z2; uu = u2
v = X2*Z1-X1Z2; vv = v2; vvv = v*vv
R = vv*X1Z2; A = uu*Z1Z2-vvv-2*R
X3 = v*A
Y3 = u*(R-A)-vvv*Y1Z2
Z3 = vvv*Z1Z2
Output: P+Q = (X3, Y3, Z3)
```

14 multiplications, 6 additions

Comparaison
Corps finis / Courbes elliptiques

Corps Finis

- 1 Multiplication de point = 1 multiplication entière
- Paramètres de sécurité pour 128 bits :
  - $L_p[1/3, 1.92]$
  - Log p = 3076

Courbes elliptiques

- 1 Multiplication de point = 14 multiplication entière
- Paramètres de sécurité pour 128 bits :
  - $L_p[1, 0.5]$
  - Log p = 256

Question :

- En considérant une complexité de multiplication en $(\log p)^2$, pour 128 bits de sécurité, quel est le ratio de performance entre corps finis/courbes elliptiques ?

Comparaison
Corps finis / Courbes elliptiques

Corps Finis

- 1 Multiplication de point = 1 multiplication entière
- Paramètres de sécurité pour 128 bits :
  - $L_p[1/3, 1.92]$
  - Log p = 3076

Courbes elliptiques

- 1 Multiplication de point = 14 multiplication entière
- Paramètres de sécurité pour 128 bits :
  - $L_p[1, 0.5]$
  - Log p = 256

Question :

- En considérant une complexité de multiplication en $(\log p)^2$, pour 128 bits de sécurité, quel est le ratio de performance entre corps finis/courbes elliptiques ?

Réponse :

- $3076^2 / (14 \cdot 256^2) = 10 \rightarrow$ Exponentiation sur les corps finis 10 fois plus lent.

Échange de clé non-interactif

Publication des clés sur un réseau social :

- A,B,C,D publient leur clé publique sur leur réseau social :
- $\boldsymbol{p_a} = g^a\,[p]$, $\boldsymbol{p_b} = g^b\,[p]$, $\boldsymbol{p_c} = g^c\,[p]$, $p_d = g^d\,[p]$

- Supposons que **A** veuille communiquer avec **C**, combien d'échange faut-il avec C pour dériver une clé secrète ?

Échange de clé non-interactif

Publication des clés sur un réseau social :

- A,B,C,D publient leur clé publique sur leur réseau social :
- $p_a = g^a [p]$, $p_b = g^b [p]$, $p_c = g^c [p]$, $p_d = g^d [p]$

- Supposons que **A** veuille communiquer avec **C**, combien d'échange faut-il avec C pour dériver une clé secrète ?
  - 0 : A récupère la clé publique de C et calcule $p_c^{\,a}$

Ce qu'il faut retenir (1)

- Diffie-Hellman (DH) est un mécanisme d'échange de clé dont la sécurité repose sur la complexité de calculer $g^{ab}[p]$ sachant (g,p,g^a,g^b).
- Le Logarithme discret permet de casser DH, c'est même la meilleure méthode connue.
- Il existe deux grandes instanciations de DH :
  - Une instanciation sur les corps finis
  - Une instanciation sur les courbes elliptiques

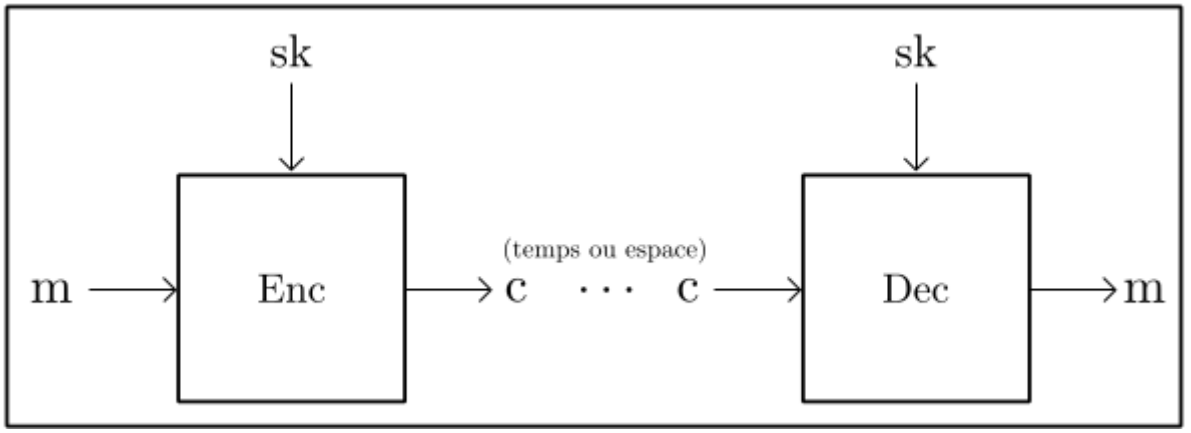
Ce qu’il faut retenir (2)

- Sous réserve d’un choix judicieux de courbe elliptique, la version basée sur les courbes elliptique possède un rapport sécurité/performance favorable.
- En particulier, la norme limite l’usage de DH sur les corps finis à un échange de clé symétrique de sécurité 112 bits.
- DH sur les courbes elliptiques peuvent attendre un partage de clé symétrique de sécurité 256 bits.

Ce qu’il faut retenir (3)

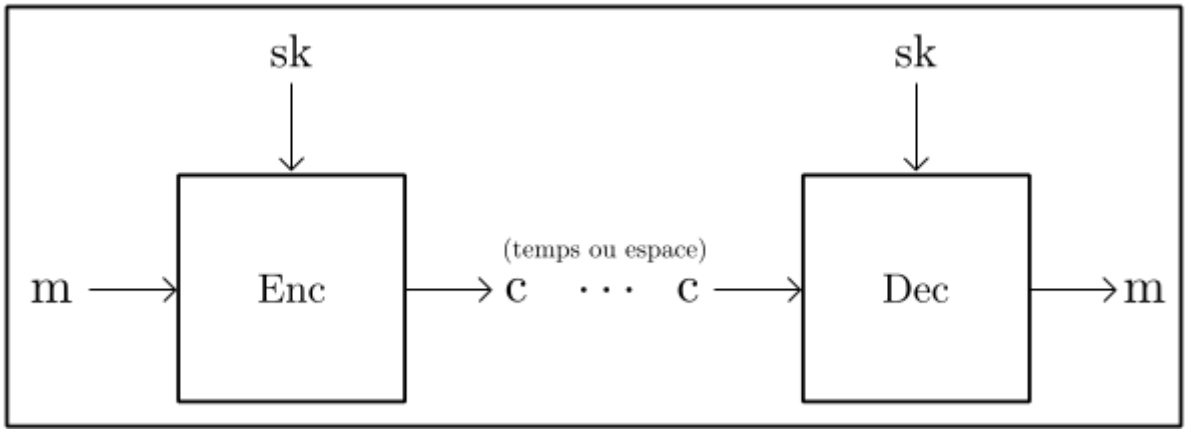
- Sans mécanisme supplémentaire, DH ne protège pas contre l’attaque du « Man-In-The-Middle », il faut prévoir une authentification de l’échange.
- Avec une mineure modification, DH permet d’attendre une confidentialité persistante (Perfect Forward Secrecy).
- Pour se faire :
  - un premier échange permet de se mettre d’accord sur une clé statique.
  - Puis, pour chaque session, un génère un secret éphémère qui est ensuite détruit.

La cryptographie asymétrique



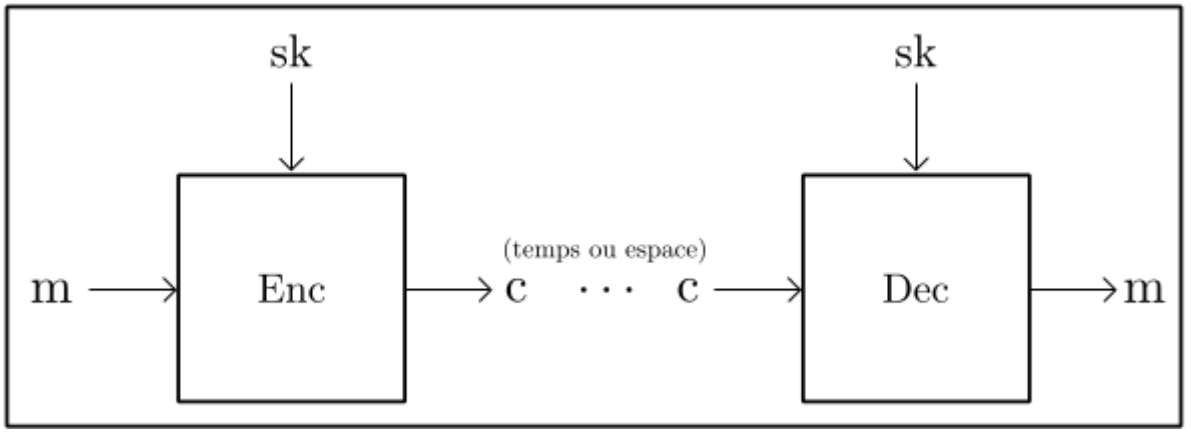
- Ce que l'on souhaite :
  - Un système réversible (chiffrement/déchiffrement) ;
  - Un système connu de tous (pour une utilisation publique) ;
  - Un système où l'on peut dévoiler la clé (de chiffrement) sans casser la sécurité.

La cryptographie asymétrique



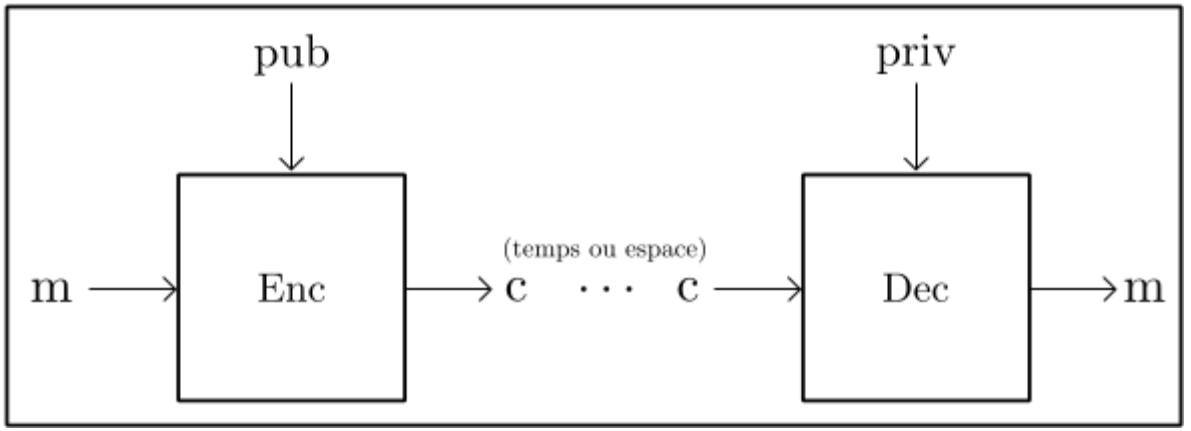
- Cas cryptographie symétrique ?
  - Un système réversible (chiffrement/déchiffrement) ;
  - Un système connu de tous (pour une utilisation publique) ;
  - Un système où l'on peut dévoiler la clé (de chiffrement) sans casser la sécurité.

La cryptographie asymétrique



- Cas cryptographie symétrique :
  - Un système réversible (chiffrement/déchiffrement) ;
  - Un système connu de tous (pour une utilisation publique) ;
  - Un système où l'on peut dévoiler la clé (de chiffrement) sans casser la sécurité.

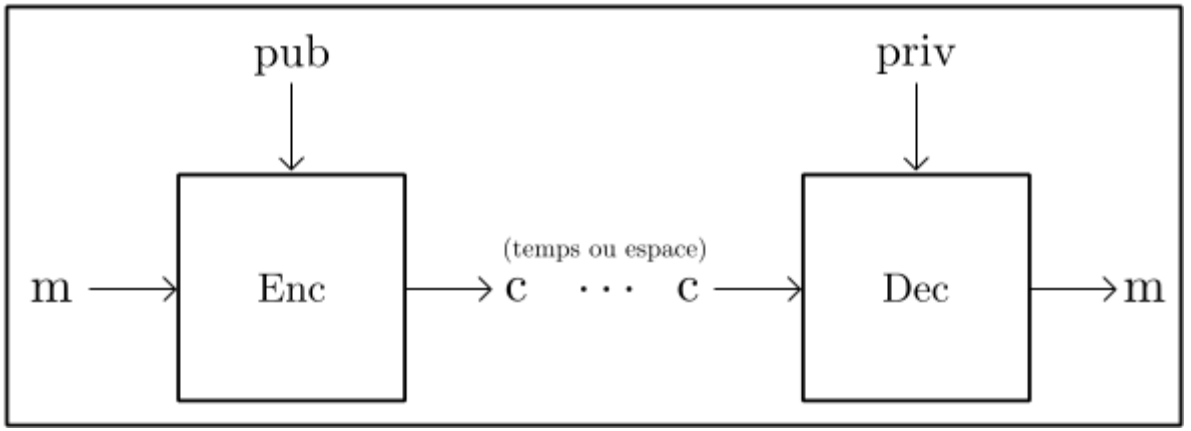
La cryptographie asymétrique



RSA (Rivest, Shamir, Adleman) (1977)

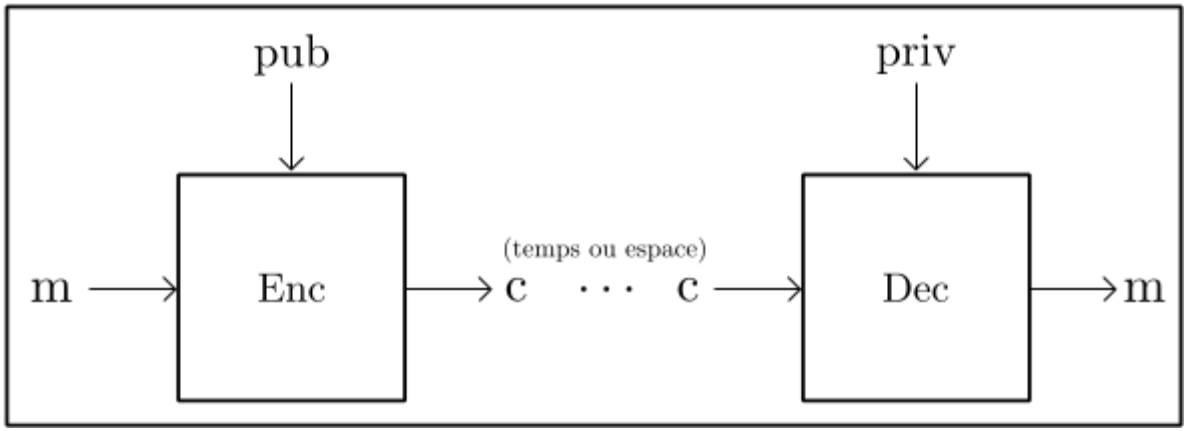
- Fonctionne avec une paire de clé publique (chiffrement), privée (déchiffrement).
- La clé publique peut être dévoilée sans impact sur la sécurité.

La cryptographie asymétrique



- Cas cryptographie asymétrique ?
  - Un système réversible (chiffrement/déchiffrement) ;
  - Un système connu de tous (pour une utilisation publique) ;
  - Un système où l'on peut dévoiler la clé (de chiffrement) sans casser la sécurité.

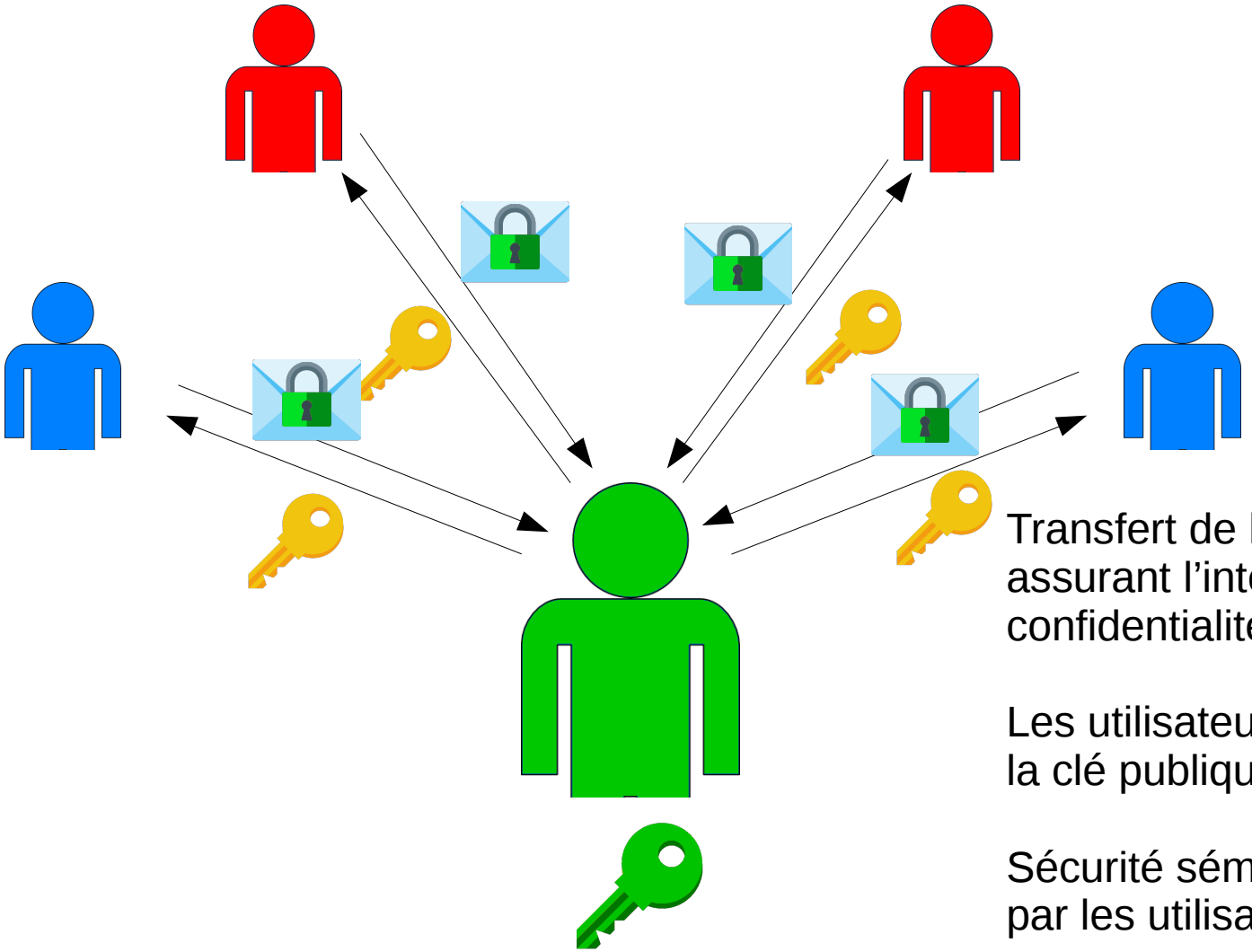
La cryptographie asymétrique



- Cas cryptographie asymétrique :
  - Un système réversible (chiffrement/déchiffrement) ;
  - Un système connu de tous (pour une utilisation publique) ;
  - Un système où l'on peut dévoiler la clé (de chiffrement) sans casser la sécurité.

Utilisation du chiffrement à clé publique

Cas 1 : Réception d'un message de n utilisateurs.



Clé secrète

Clé publique

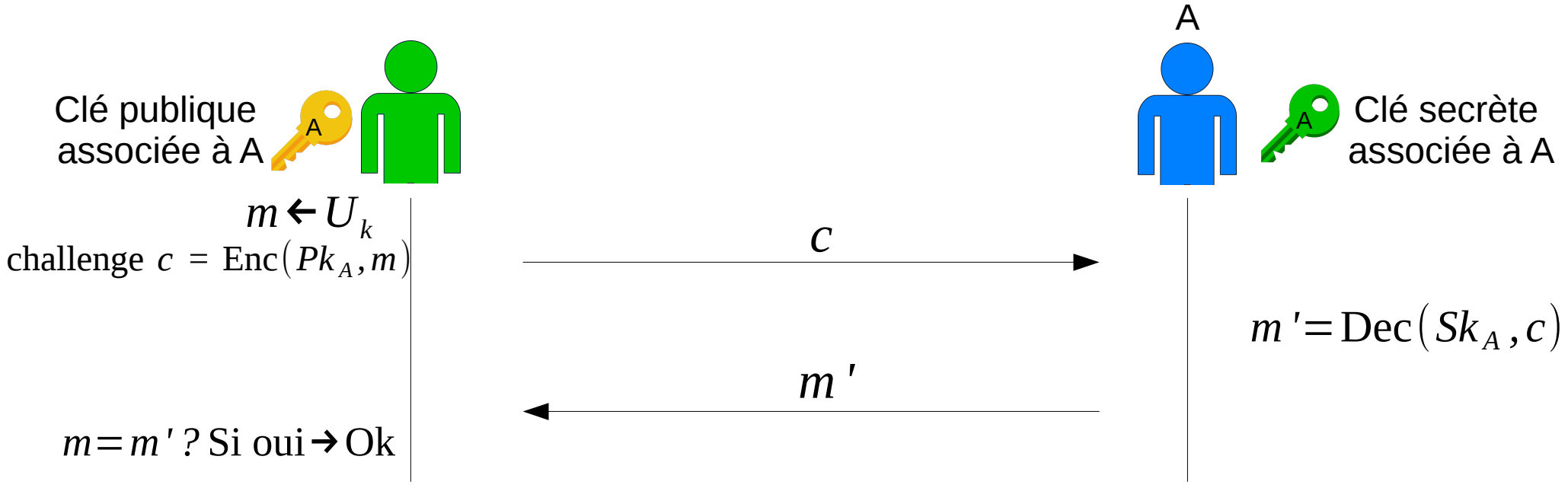
Transfert de la clé publique sur un canal assurant l'intégrité. Pas de nécessité de confidentialité.

Les utilisateurs chiffrent leur message avec la clé publique.

Sécurité sémantique sur le message chiffré par les utilisateurs.

Utilisation du chiffrement à clé publique

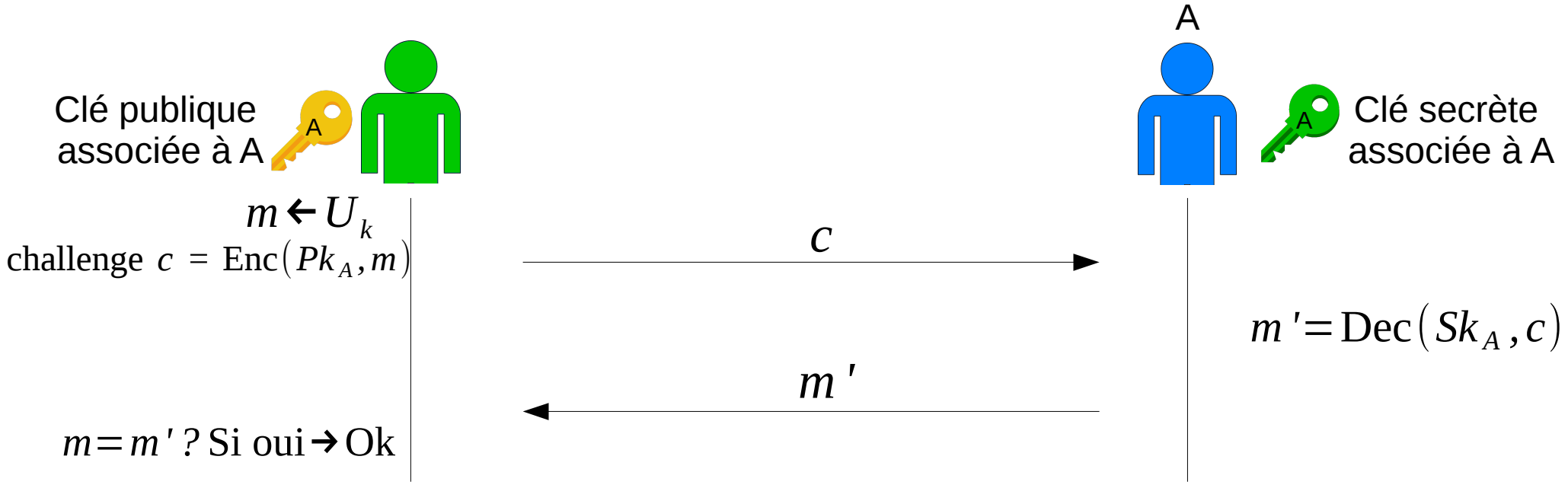
Cas 2 : Authentification



Coté sécurité, il ne faut pas oublier que la taille de m intervient : min (k, k_{PKC})

Utilisation du chiffrement à clé publique

Cas 2 : Authentification

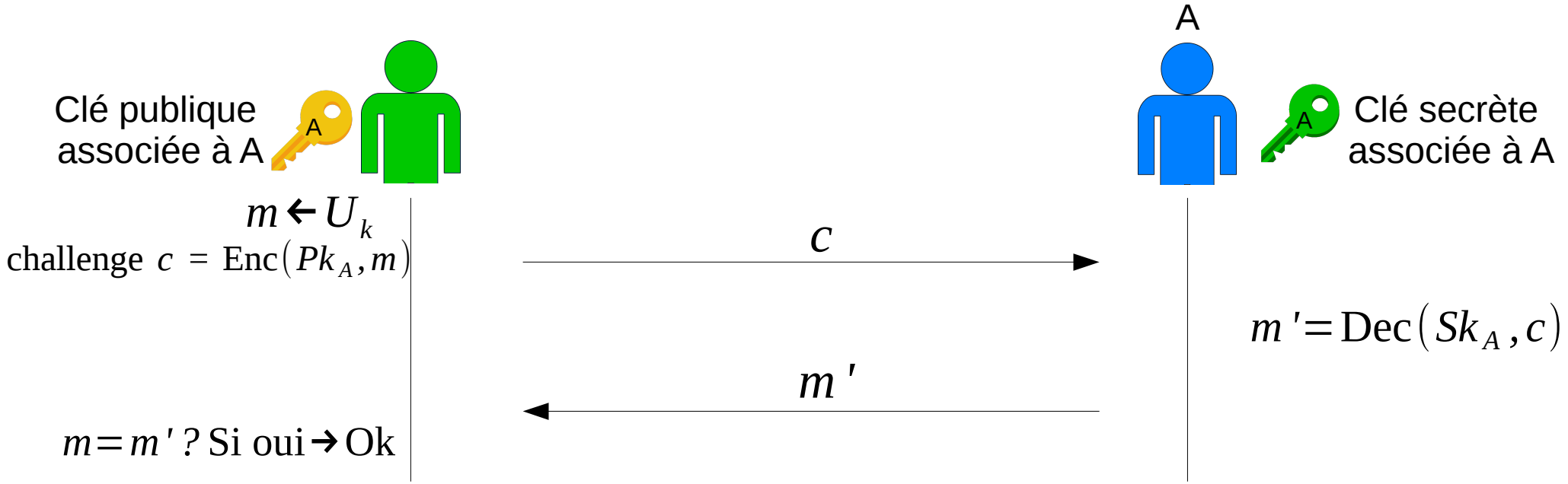


Question :

- A votre avis, quel est l'avantage du chiffrement asymétrique par rapport à un mot de passe ?

Utilisation du chiffrement à clé publique

Cas 2 : Authentification



Question :

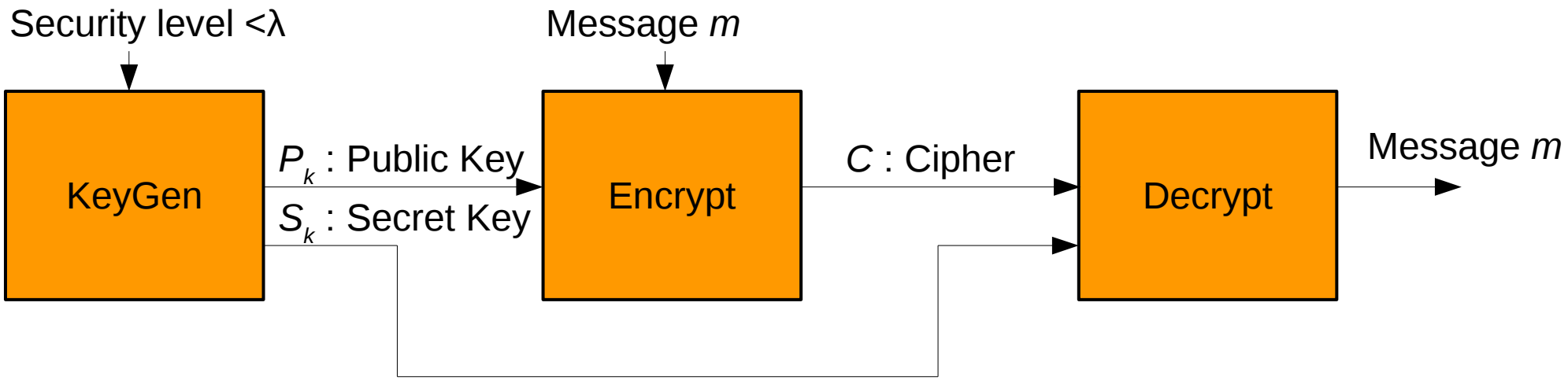
- A votre avis, quel est l'avantage du chiffrement asymétrique par rapport à un mot de passe ?

(Élément) de réponse :

- Vous n'avez pas à fournir votre secret à votre interlocuteur

Chiffrement symétrique (ou a clé publique) : PKC

- Définition classique des algorithmes :



- Encrypt ≠ Crypter
- Decrypt ≠ Decrypter = Casser le message chiffré

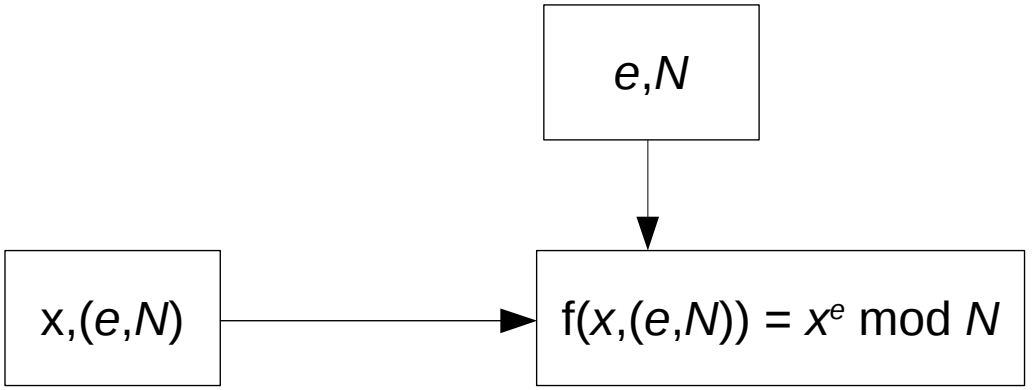
$$\forall(\lambda,m),(P_k,S_k)=\mathrm{KeyGen}(1^{\lambda}),m=\mathrm{Decrypt}(\mathrm{Encrypt}(m,P_k),S_k)$$

Chiffrement symétrique : Sécurité

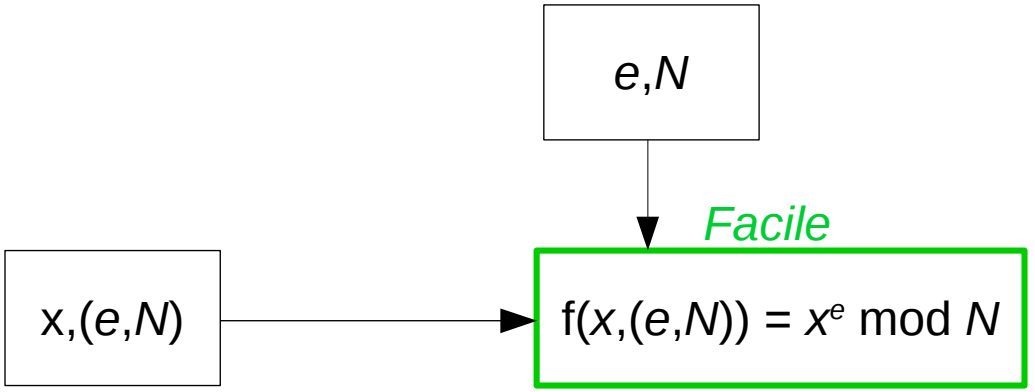
Les chiffrements symétriques sont construits avec une fonction à Trappe

- $(P_k, S_k) \leftarrow \text{KeyGen}(1^\lambda)$
- $f : (x, P_k) \rightarrow f(x, P_k)$
- Sachant $f(x, P_k)$ et P_k, trouver x est ***difficile*** (en temps polynomial).
- Sachant $f(x, P_k)$ et S_k, Trouver x est ***facile***

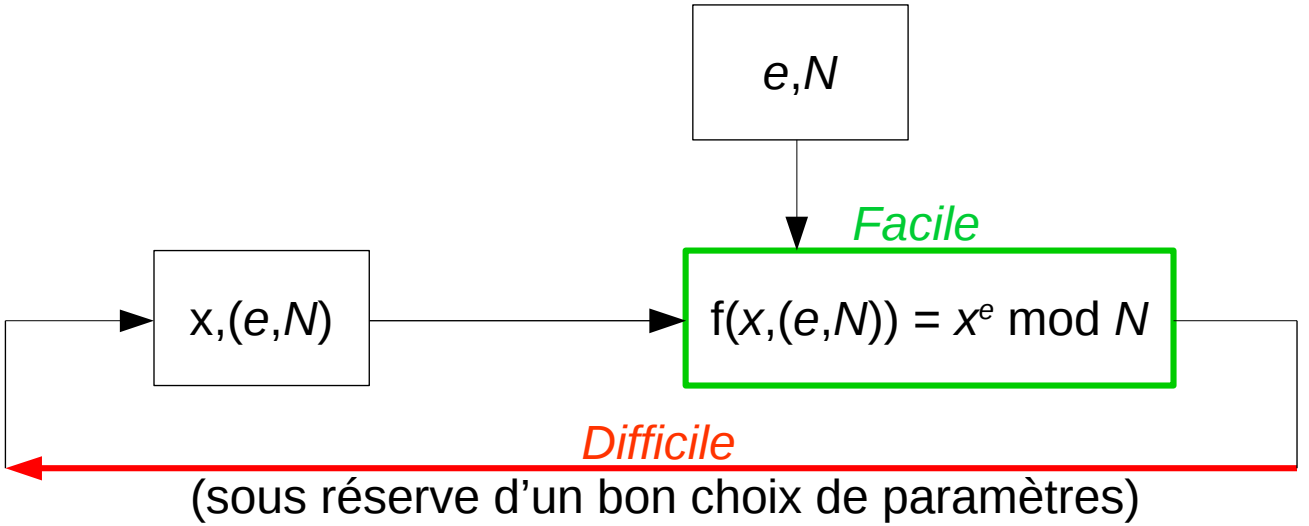
Textbook RSA



Textbook RSA

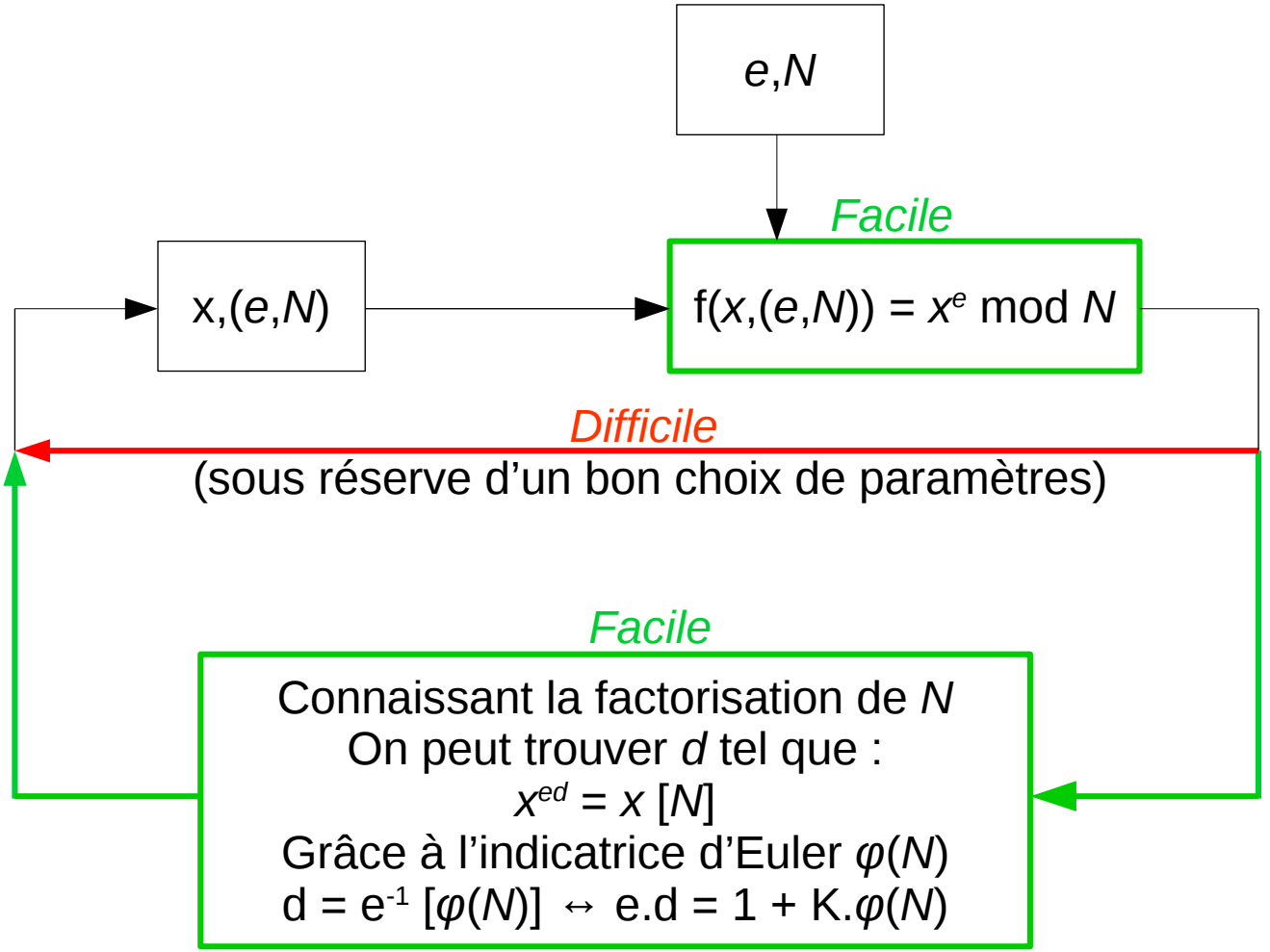


Textbook RSA

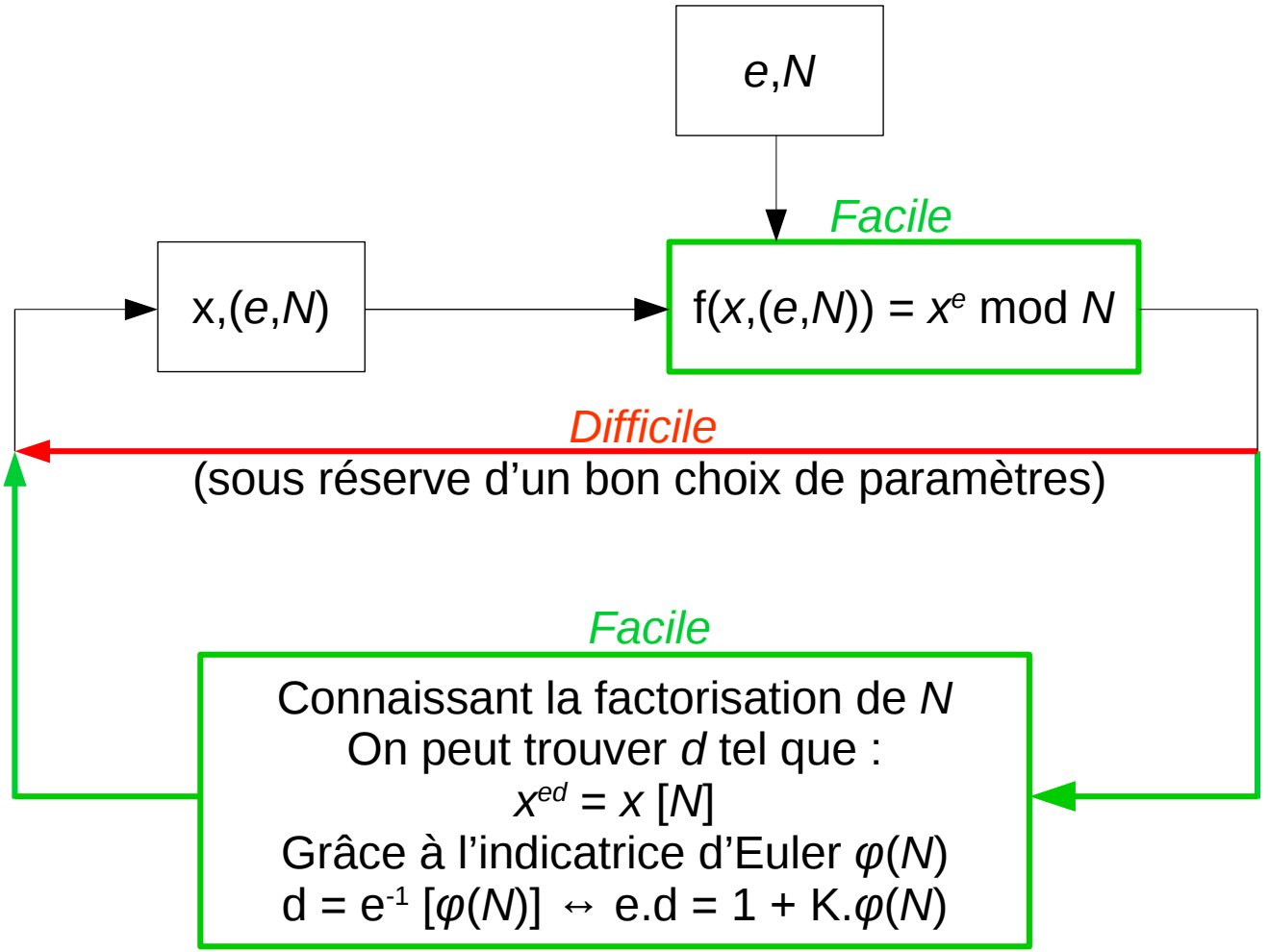


(sous réserve d'un bon choix de paramètres)

Textbook RSA



Textbook RSA



$\varphi(N)$ dénombre tous les entiers entre 1 et N -1 premiers avec N

Cas de RSA :

On prends deux grands nombres premiers p et q, on choisi $N = p \cdot q$

$\varphi(N) = (p-1).(q-1)$

Limitations de *Textbook* RSA

Modification du message par un attaquant

Question :

L'attaquant récupère $C = m^e$ [N]. Peut-il modifier le message contenu dans C ?

Limitations de *Textbook* RSA

Modification du message par un attaquant

Question :

L'attaquant récupère $C = m^e$ [N]. Peut-il modifier le message contenu dans C ?

Réponse :

Oui, et simplement :

$C' = 2^e.m^e$ [N] → Decrypt(C') = $2.m$

Il faut donc ajouter un mécanisme permettant de détecter la modification du message (authentification).

Limitations de *Textbook* RSA

Détermination du message connaissant la forme de celui-ci

Question :

L’attaquant récupère un message chiffré. Il sait que le message déchiffré est soit « Transaction acceptée », soit « Transaction refusée ». Peux-t-il récupérer simplement le message sans le modifier ?

Limitations de *Textbook* RSA

Détermination du message connaissant la forme de celui-ci

Question :

L'attaquant récupère un message chiffré. Il sait que le message déchiffré est soit « Transaction acceptée », soit « Transaction refusée ». Peux-t-il récupérer simplement le message sans le modifier ?

Réponse :

Oui, il chiffre les deux cas possibles, et compare avec le message chiffré. On dit que *Textbook* RSA est déterministe.

Il faut donc randomiser le message.

Authentification/randomisation de *Textbook* RSA

Authentification + randomisation

→ Fonction de hachage + source d’aléa

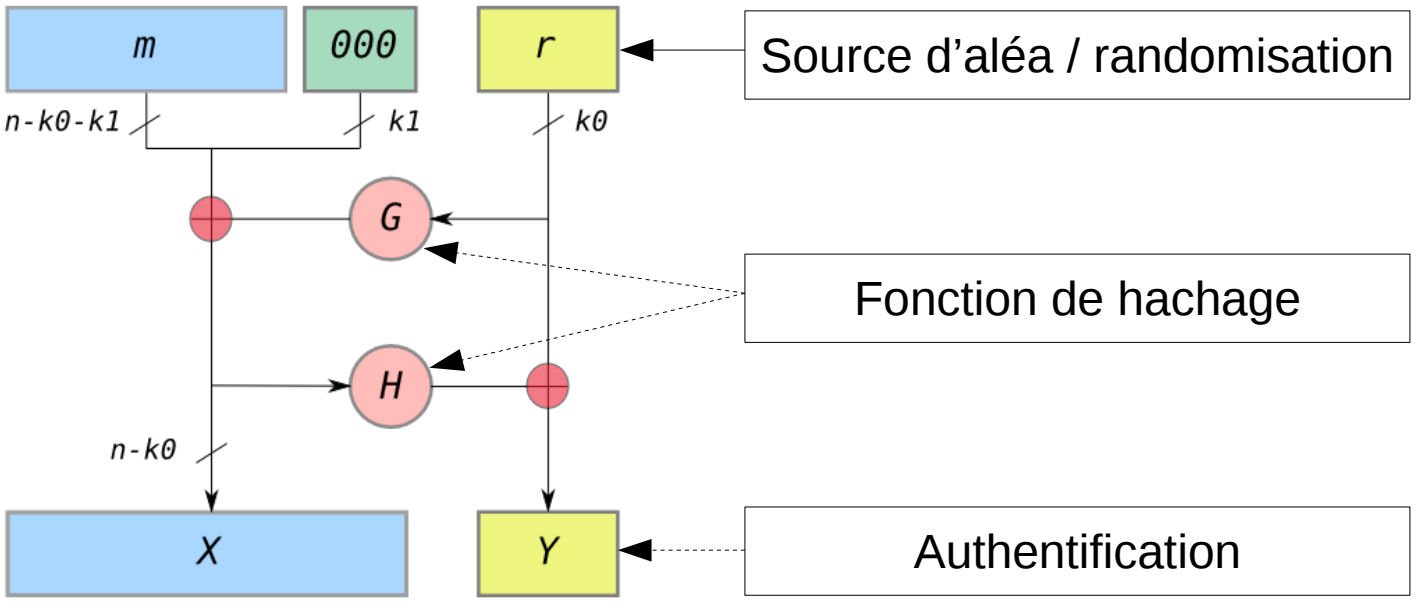
Authentification/randomisation de *Textbook* RSA

Authentification + randomisation

→ Fonction de hachage + source d'aléa

Méthode standardisée 1 (PKCS#1) : (Historique mais secure)

Optimal Asymmetric Encryption Padding (OAEP)

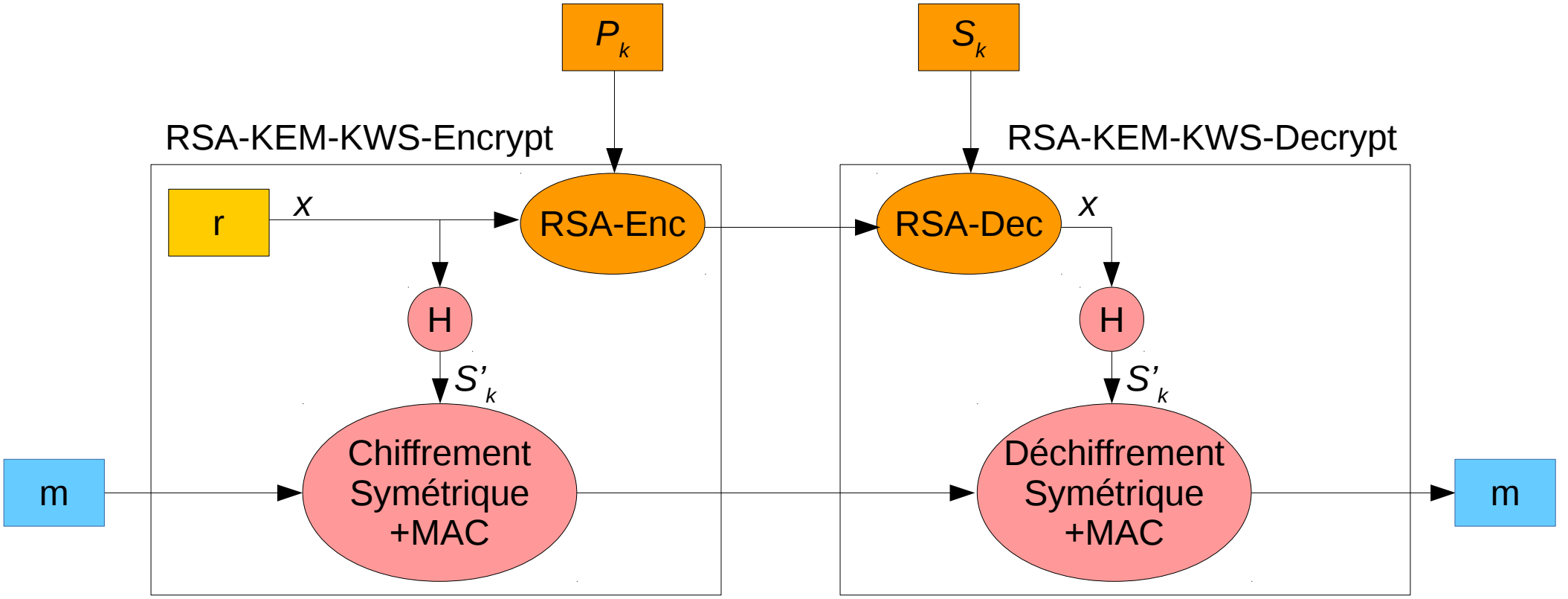


Authentification/randomisation de *Textbook* RSA

Méthode standardisée 2 (NIST SP 800 56B) :

(mécanisme à portée plus large que RSA)

Mécanisme d'encapsulation

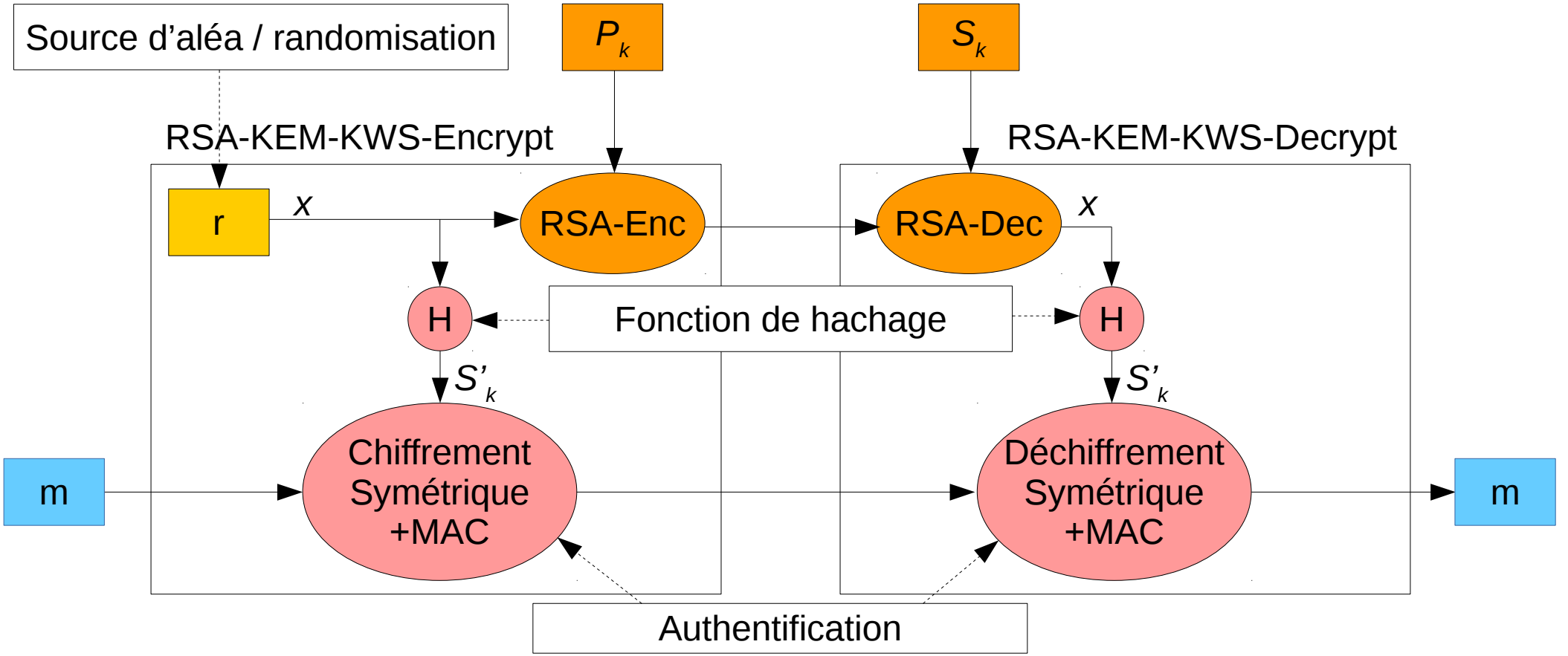


Authentification/randomisation de *Textbook* RSA

Méthode standardisée 2 (NIST SP 800 56B) :

(mécanisme à portée plus large que RSA)

Mécanisme d'encapsulation



Principe de cryptanalyse sous-jacent

Évaluation de la sécurité par classe d'attaque

La sécurité est standardisée par les *classes d'attaque*, donnant un aperçu des capacités d'un attaquant :

- Attaque sur chiffré seul
- Attaque à clair connu (Known Plaintext Attack)
- Attaque à clair choisi (Chosen Plaintext Attack)
- Attaque à chiffré choisi (Chosen Ciphertext Attack)

Principe de cryptanalyse sous-jacent

Évaluation de la sécurité par classe d'attaque

La sécurité est standardisée par les *classes d'attaque*, donnant un aperçu des capacités d'un attaquant :

- Attaque sur chiffré seul
  - L'attaquant base son attaque sur seulement les messages chiffrés.
- Attaque à clair connu (Known Plaintext Attack)
- Attaque à clair choisi (Chosen Plaintext Attack)
- Attaque à chiffré choisi (Chosen Ciphertext Attack)

Principe de cryptanalyse sous-jacent

Évaluation de la sécurité par classe d'attaque

La sécurité est standardisée par les *classes d'attaque*, donnant un aperçu des capacités d'un attaquant :

- Attaque sur chiffré seul
- Attaque à clair connu (Known Plaintext Attack)
  - L'attaque se base sur des couples message ↔ chiffré.
- Attaque à clair choisi (Chosen Plaintext Attack)
- Attaque à chiffré choisi (Chosen Ciphertext Attack)

Principe de cryptanalyse sous-jacent

Évaluation de la sécurité par classe d'attaque

La sécurité est standardisée par les *classes d'attaque*, donnant un aperçu des capacités d'un attaquant :

- Attaque sur chiffré seul
- Attaque à clair connu (Known Plaintext Attack)
- Attaque à clair choisi (Chosen Plaintext Attack)
  - L'attaquant à les moyens de chiffrer les messages. C'est le cas du chiffrement à clé publique, donc c'est la sécurité minimale à avoir.
- Attaque à chiffré choisi (Chosen Ciphertext Attack)

Principe de cryptanalyse sous-jacent

Évaluation de la sécurité par classe d'attaque

La sécurité est standardisée par les *classes d'attaque*, donnant un aperçu des capacités d'un attaquant :

- Attaque sur chiffré seul
- Attaque à clair connu (Known Plaintext Attack)
- Attaque à clair choisi (Chosen Plaintext Attack)
- Attaque à chiffré choisi (Chosen Ciphertext Attack)
  - L'attaquant peut également demander à ce que l'on lui déchiffre les messages chiffrés. Si l'attaquant peut faire une attaque CCA, le cryptosystème est potentiellement dangereux si l'attaquant est actif.

Principe de cryptanalyse sous-jacent

- Ainsi, les méthodes décrites plus haut permettent de passer RSA d'un niveau de sécurité **IND-CPA** (Chosen Plaintext Attack), à **IND-CCA** (Chosen Ciphertext Attack).
- Elles nous apportent un certain niveau de sécurité contre une attaque dynamique, et nous avons vu quelles étaient réalistes dans 2 cas de figure (modification du message, déchiffrement par apprentissage).

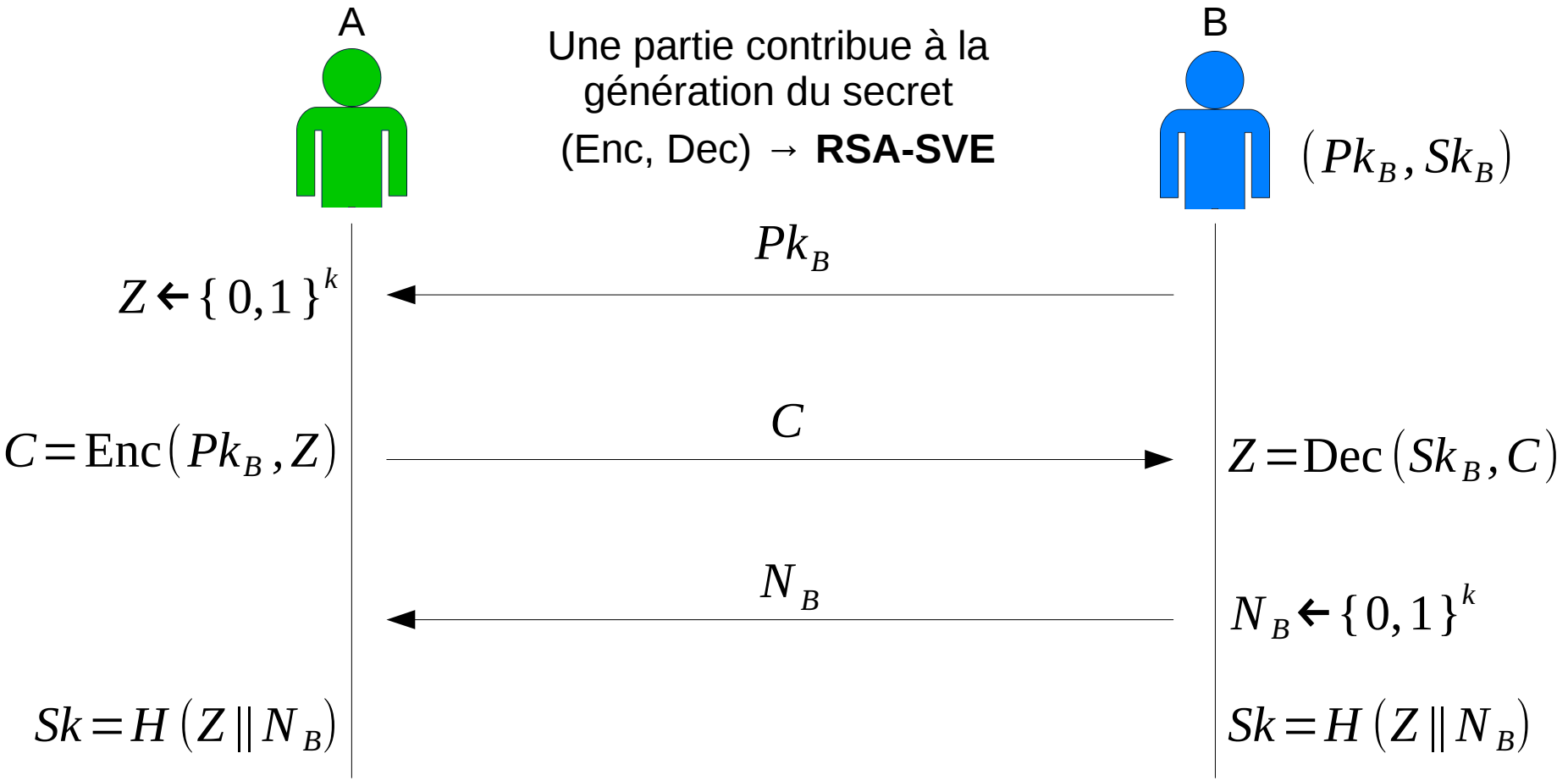
Constructions proposées par le standard NIST SP 800 56B

Recommandation pour l'échange de clé basé sur le problème de factorisation d'entiers (i.e. RSA)

Deux méthodes sont décrites :

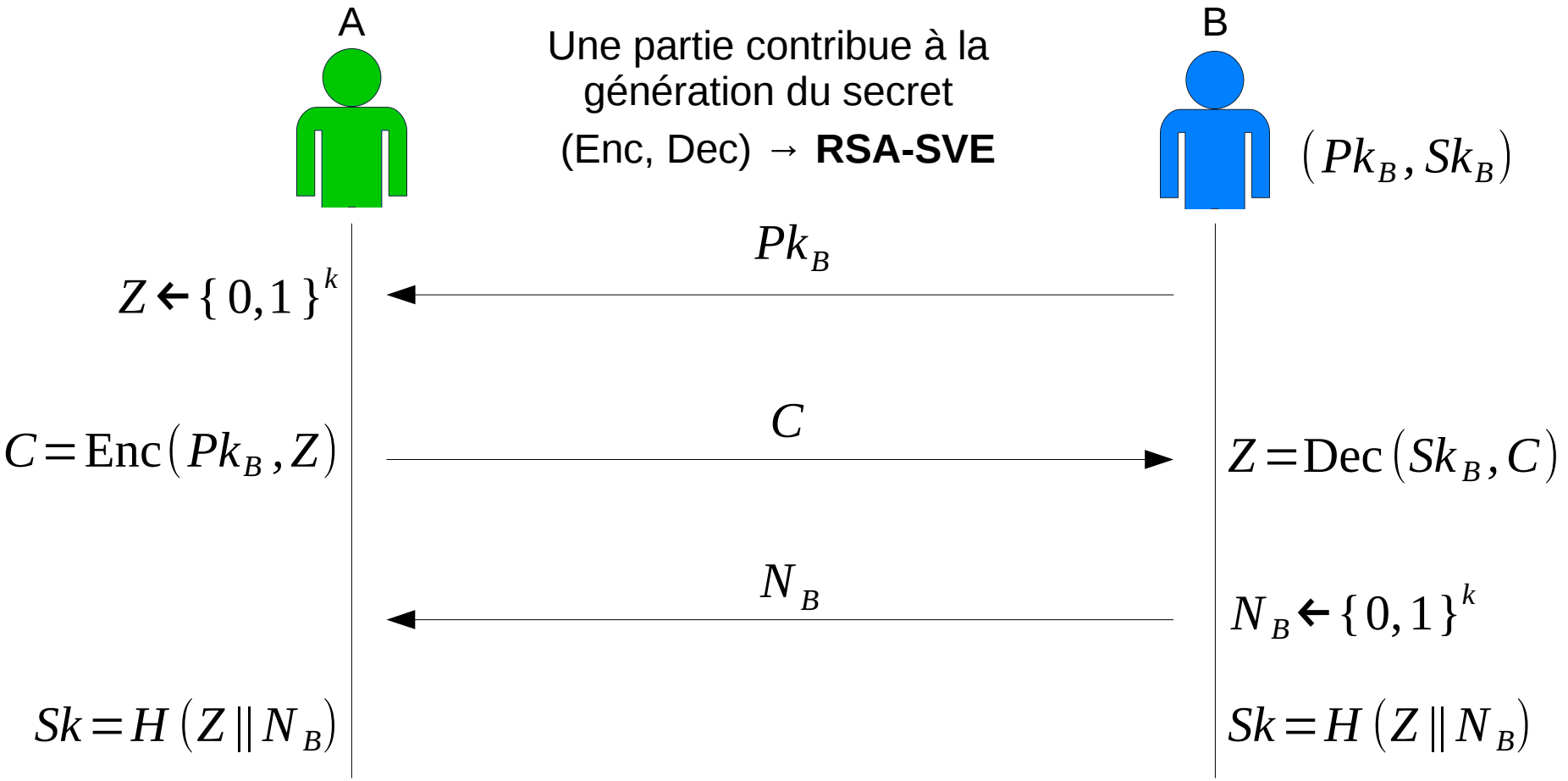
- **Schémas d'accord de clé**
  (Key Agreement Schemes KAS1 et KAS2)
  Les deux parties contribuent au secret
  - RSA-SVE : $x \leftarrow \{0,1\}^n$, Enc(P_k,x)
- **Shémas de transport de clé**
  (Key Transport Schemes)
  - RSA-KEM-KWS
  - RSA-OAEP

Accords de clés : KAS1, KAS2



Le Nonce est une donnée publique, à votre avis à quoi sert-il ?

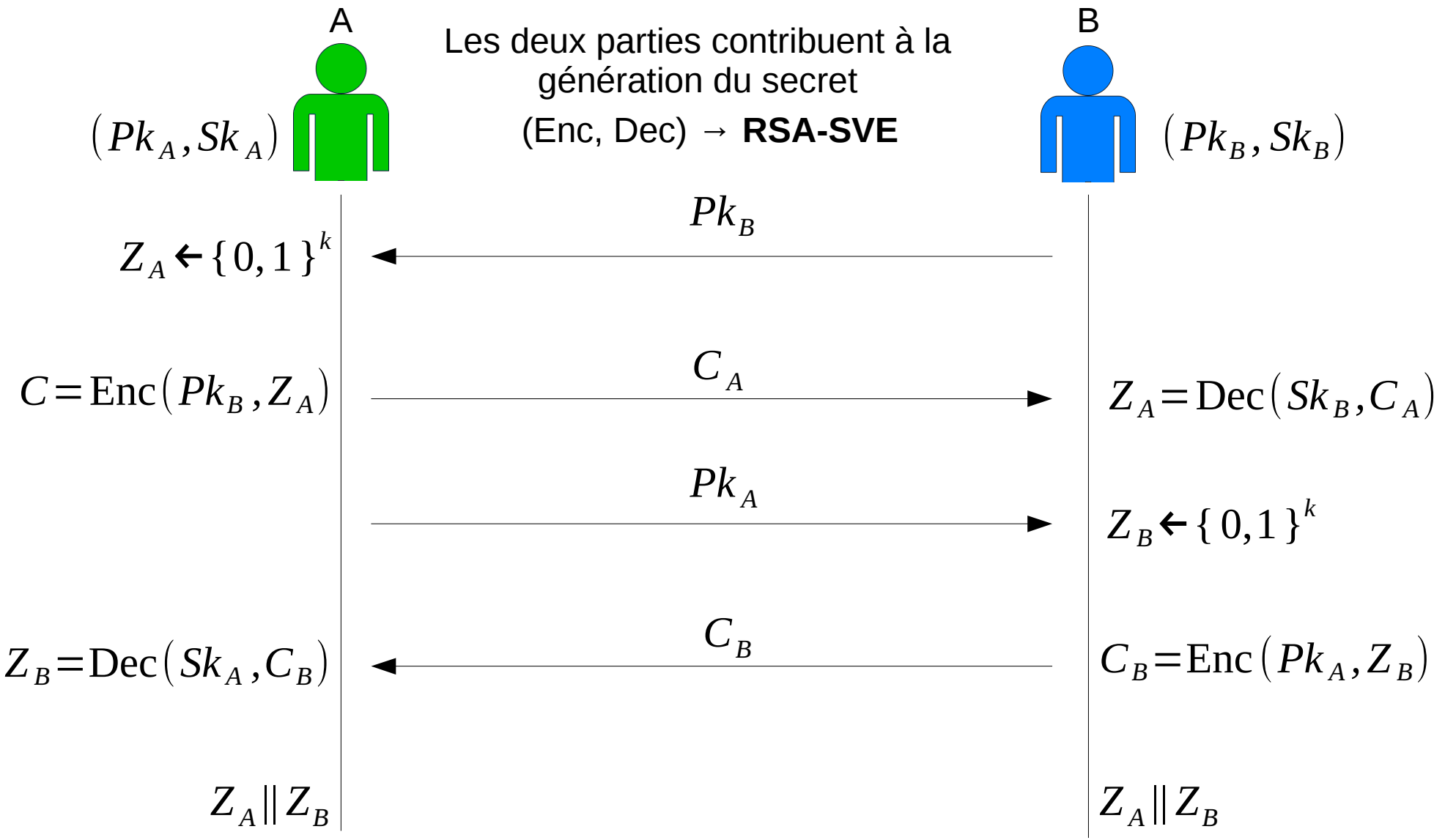
Accords de clés : KAS1, KAS2



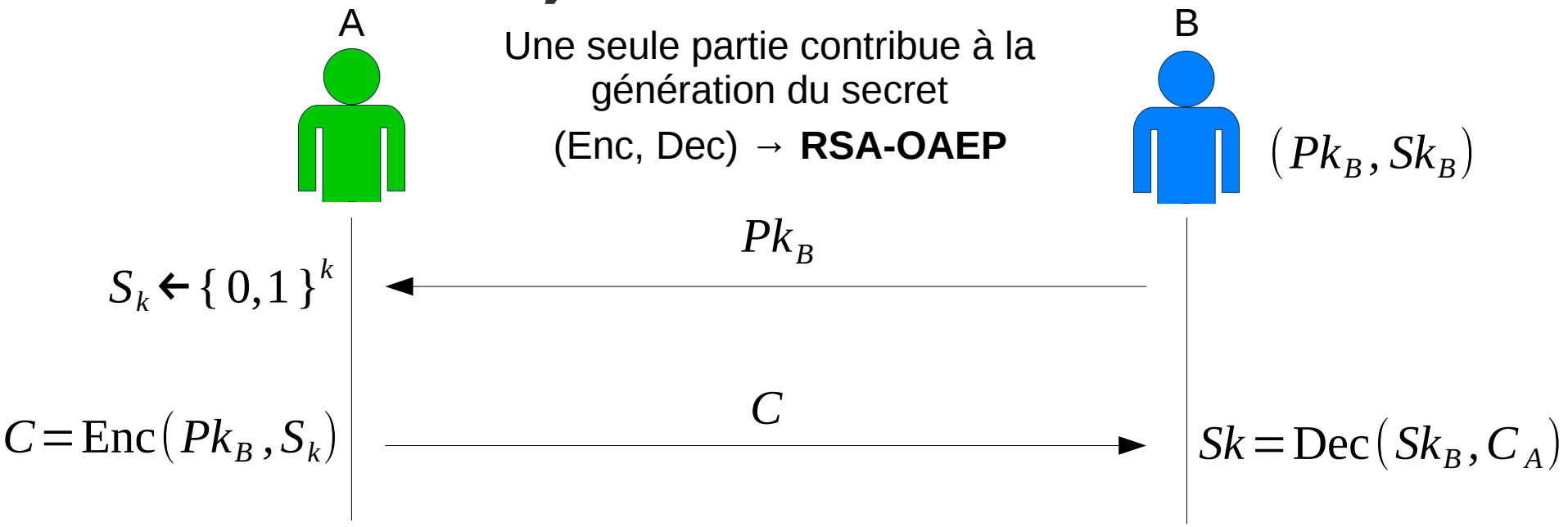
Le Nonce est une donnée publique, à votre avis à quoi sert-il ?

C'est l'équivalent de l'identifiant de la connexion.

Accords de clés : KAS1, KAS2

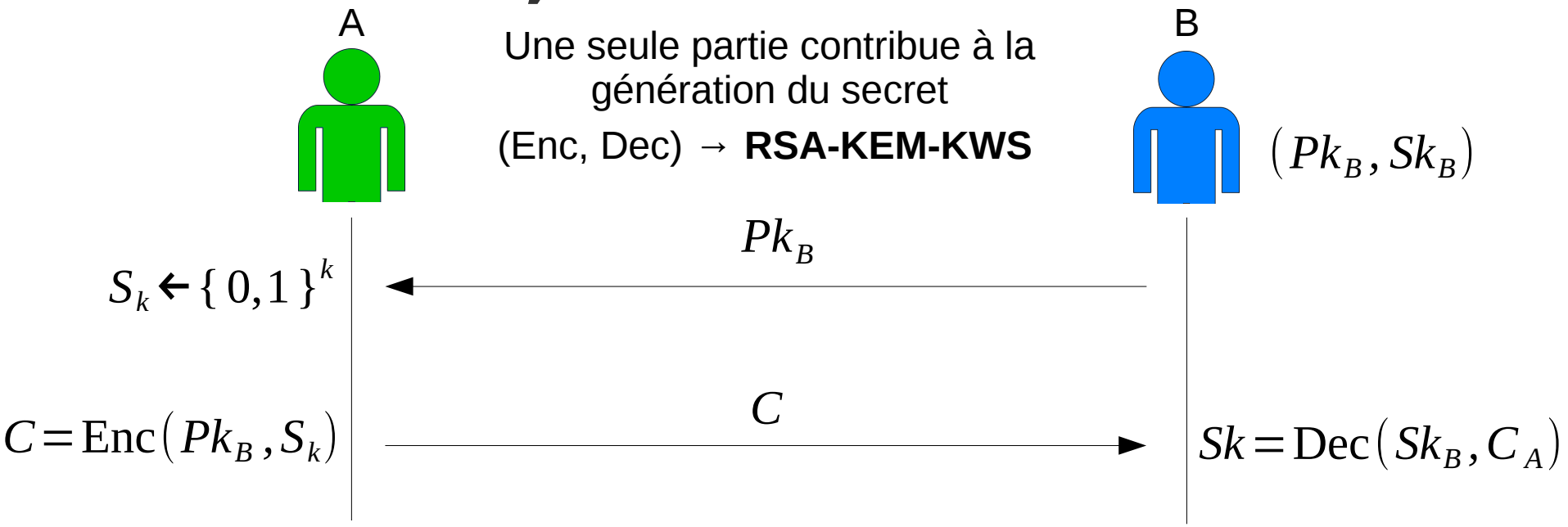


Transports de clés : KTS-OAEP, KTS-KEM-KWS



RSA-OAEP est déjà IND-CCA2, le transport de clé est donc minimal.

Transports de clés : KTS-OAEP, KTS-KEM-KWS

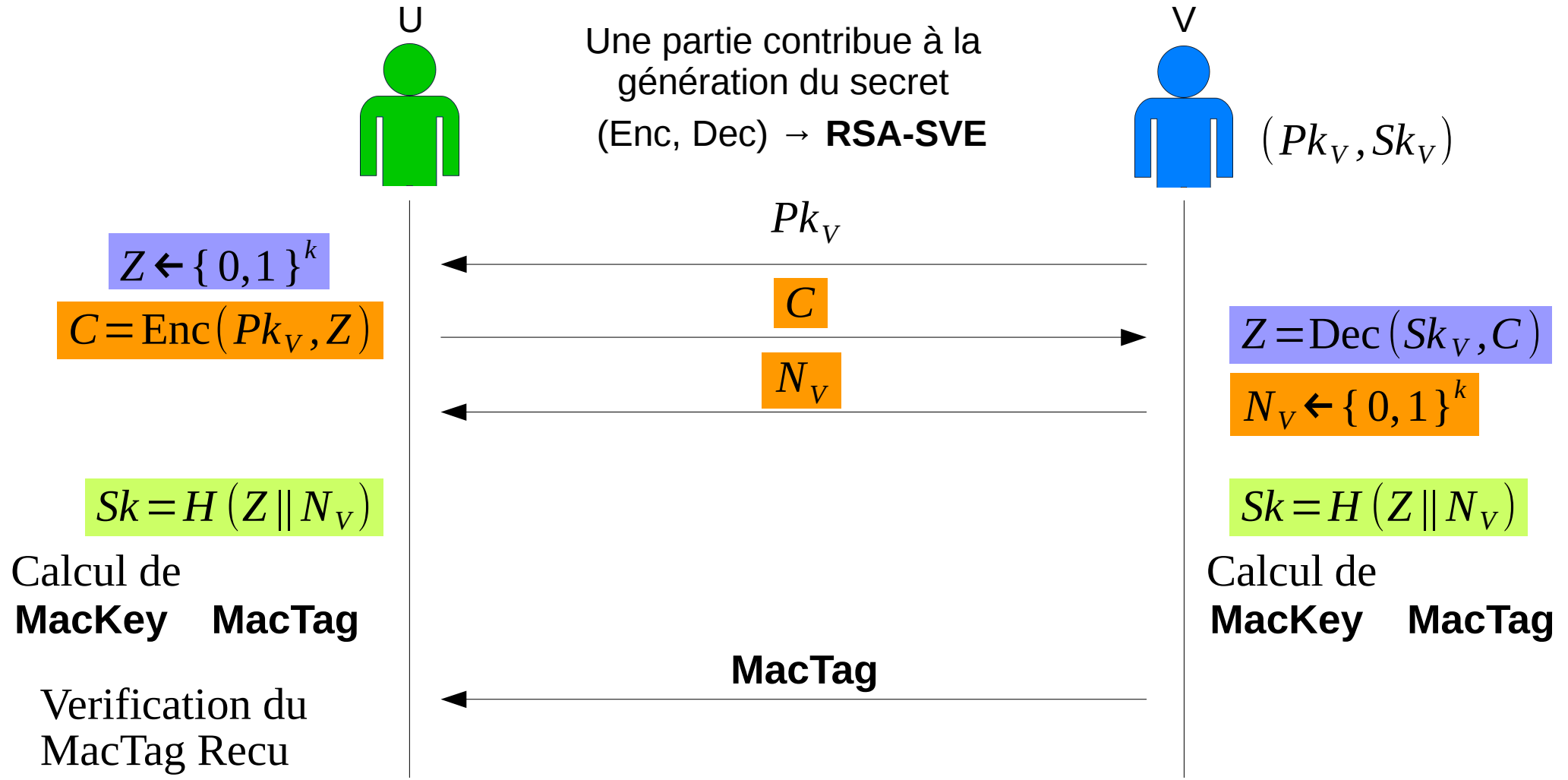


RSA-KEM-KWS est déjà IND-CCA2, le transport de clé est donc minimal.

La confirmation de clé

- C’est une étape facultative qui permet de ***s’assurer*** que notre interlocuteur a bien ***reçu la clé***.
- Pour se faire il faut trouver un moyen de ***dériver du secret une valeur*** permettant de ***confirmer*** le bon calcul, tout en ne donnant pas d’information sur le secret.
- Cette clé est ***associée à un binding***, c’est-à-dire des ***informations spécifiques à la transaction actuelle***. Exemples :
  - Date ;
  - Algorithmes utilisés ;
  - Informations publiques des parties ;
  - Information secrète des parties.

Exemple d'accords de clés KAS1 avec confirmation



MacKey = H (1 || ***Z*** || Contexte) || H (2 || ***Z*** || Contexte) || H (3 || ***Z*** || Contexte) …

MacTag = HMAC (**MacKey**, **KC_1_V** || **Idv** || **Idu** || **Nv** || **C**) …

Données éphémères | Secret Partagé | Graine secrète | Clé symétrique pour la confirmation

Les algorithmes de dérivation de clé (KDF) en une étape

KDF1/MGF1 (ISO 18033-2, IEEE P1363A, PKCS#1 v2)

CléDérivée = **H** (Z || 0 || Contexte) || **H** (Z || 1 || Contexte) **H** (Z || 2 || Contexte) || …

= **H** (Z || **C** || Contexte) || … , **C** compteur commençant à 0.

KDF2 (ISO 18033-2, ANSI X9.42)

CléDérivée = **H** (Z || **C** || Contexte) || … , **C** compteur commençant à 1.

KDF3 (ISO 18033-2, NIST SP 800 56A/B)

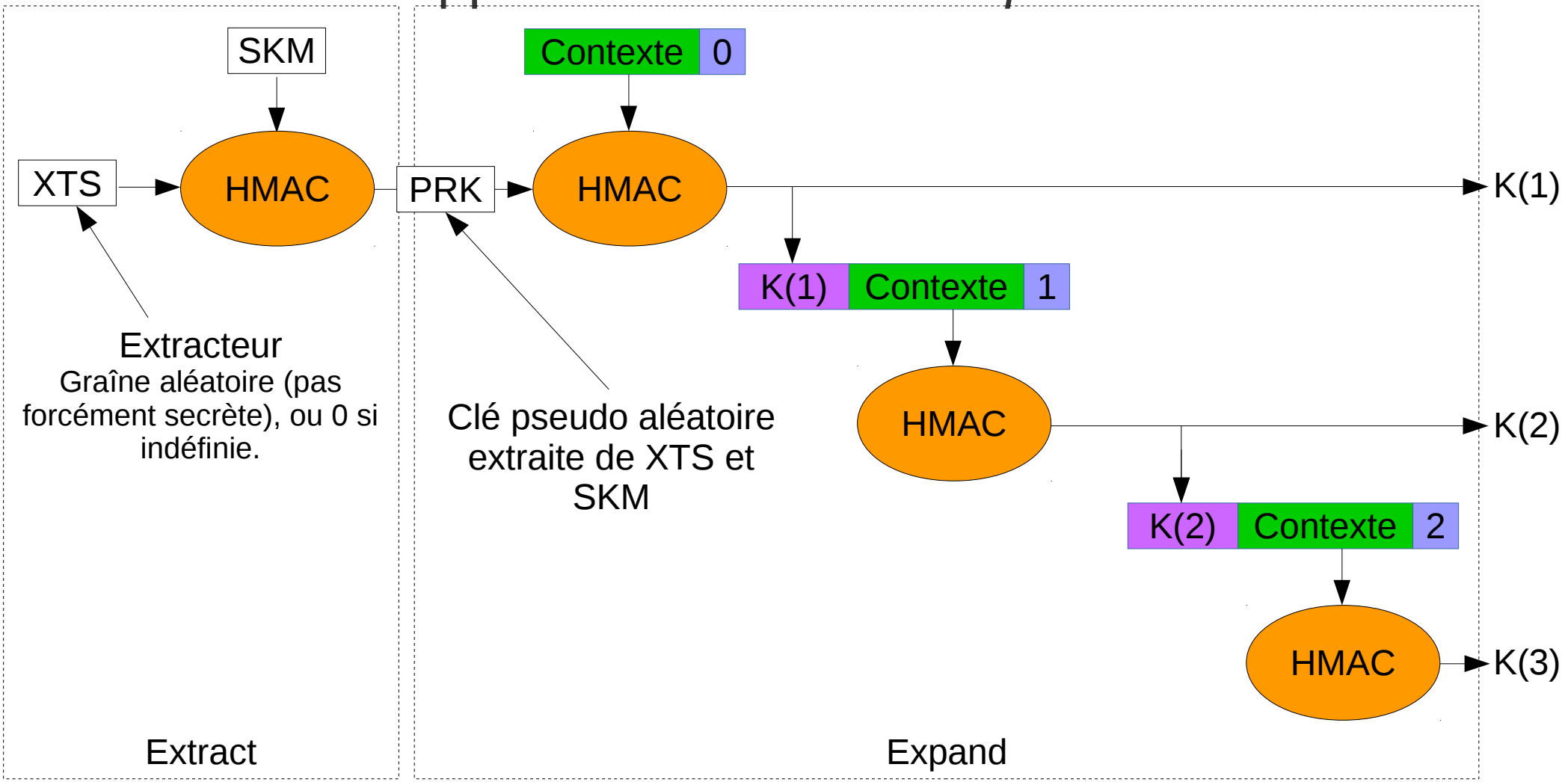
CléDérivée = **H** (**C** || Z || Contexte) || … , **C** compteur commençant à 0.

KDF3 (aternative NIST SP 800 56A/B)

CléDérivée = **HMAC** (salt , **C** || Z || Contexte) || … , **C** compteur commençant à 0, salt = 0 si non définit autre part.

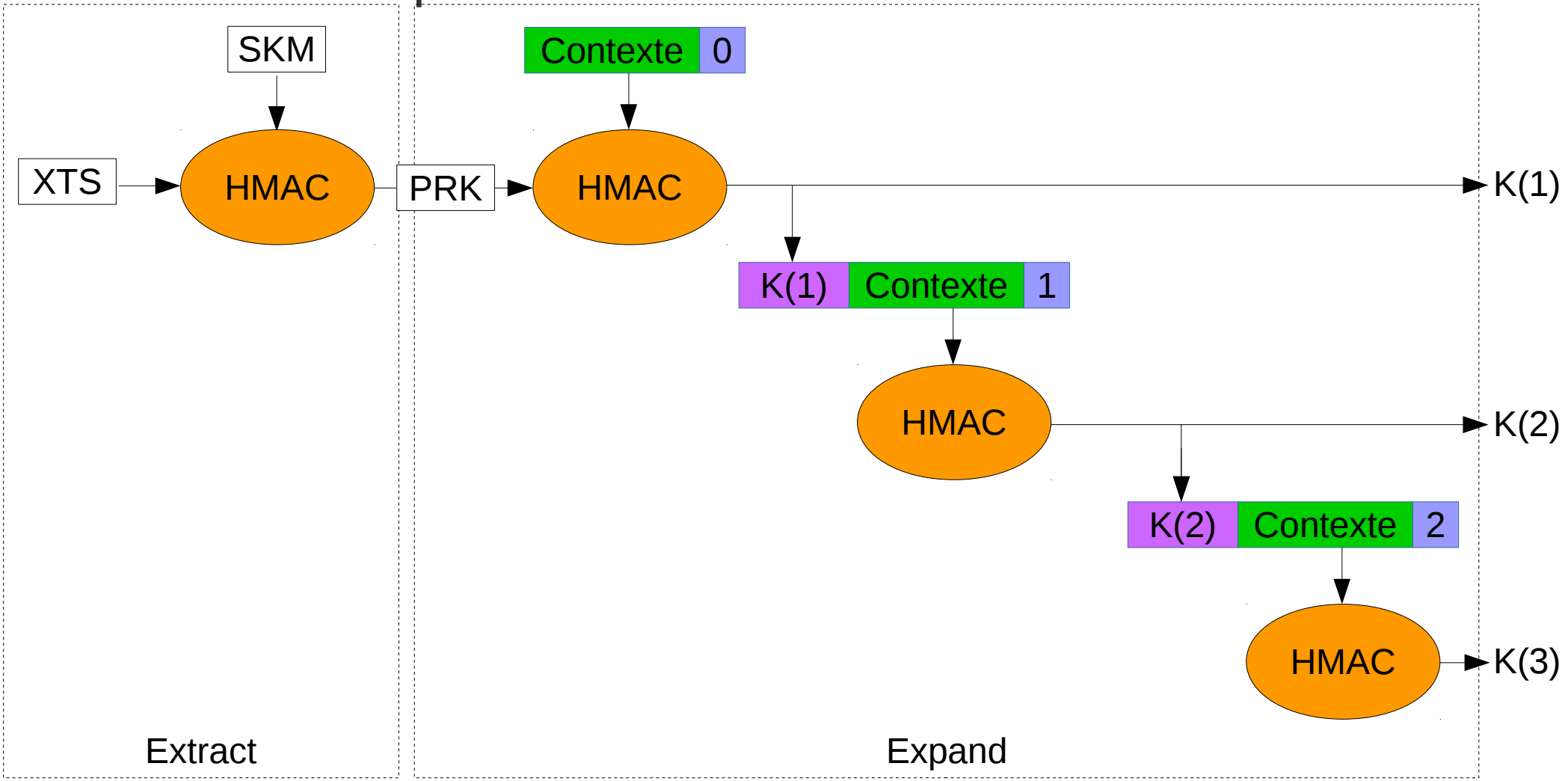
Les algorithmes de dérivation de clé (KDF) en deux étapes

Méthode appelée *Extract-then-Expand*



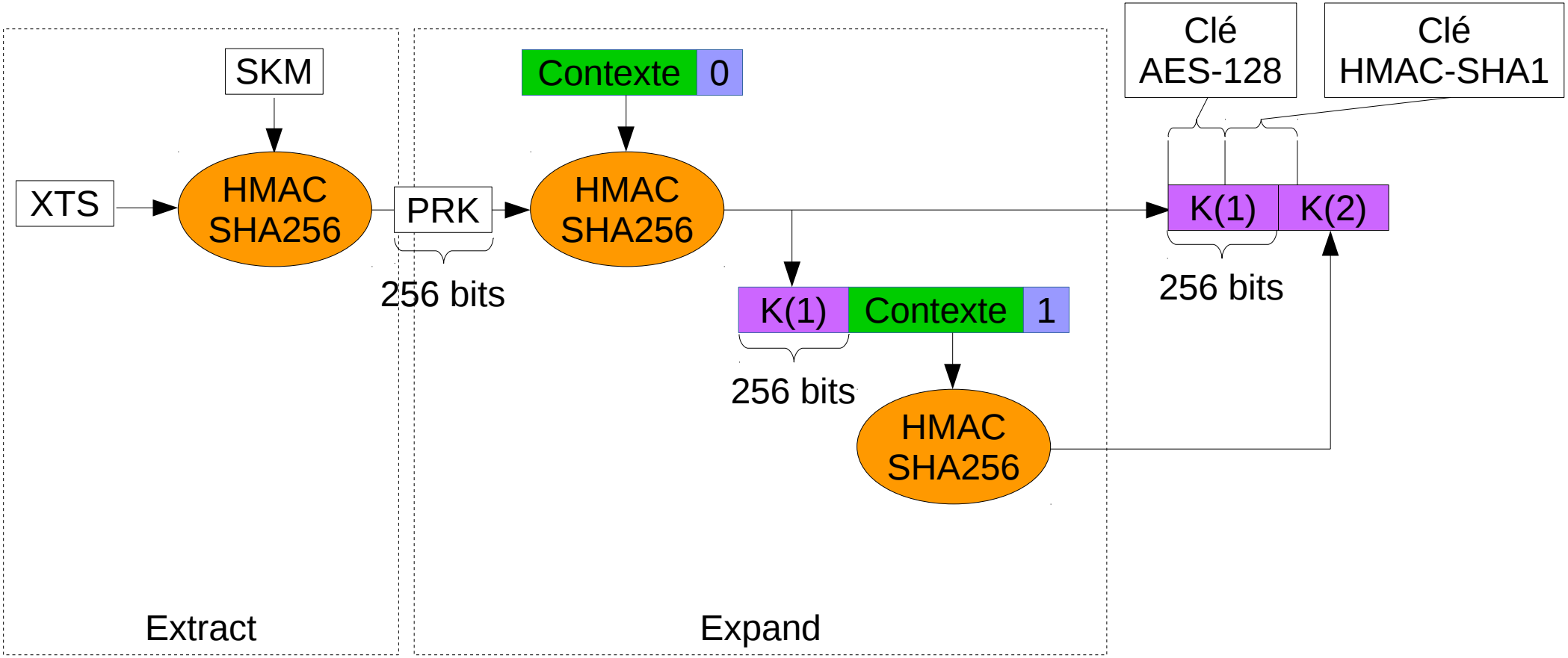
Les algorithmes de dérivation de clé (KDF) en deux étapes

Contexte permet d'associer la dérivation à un contexte



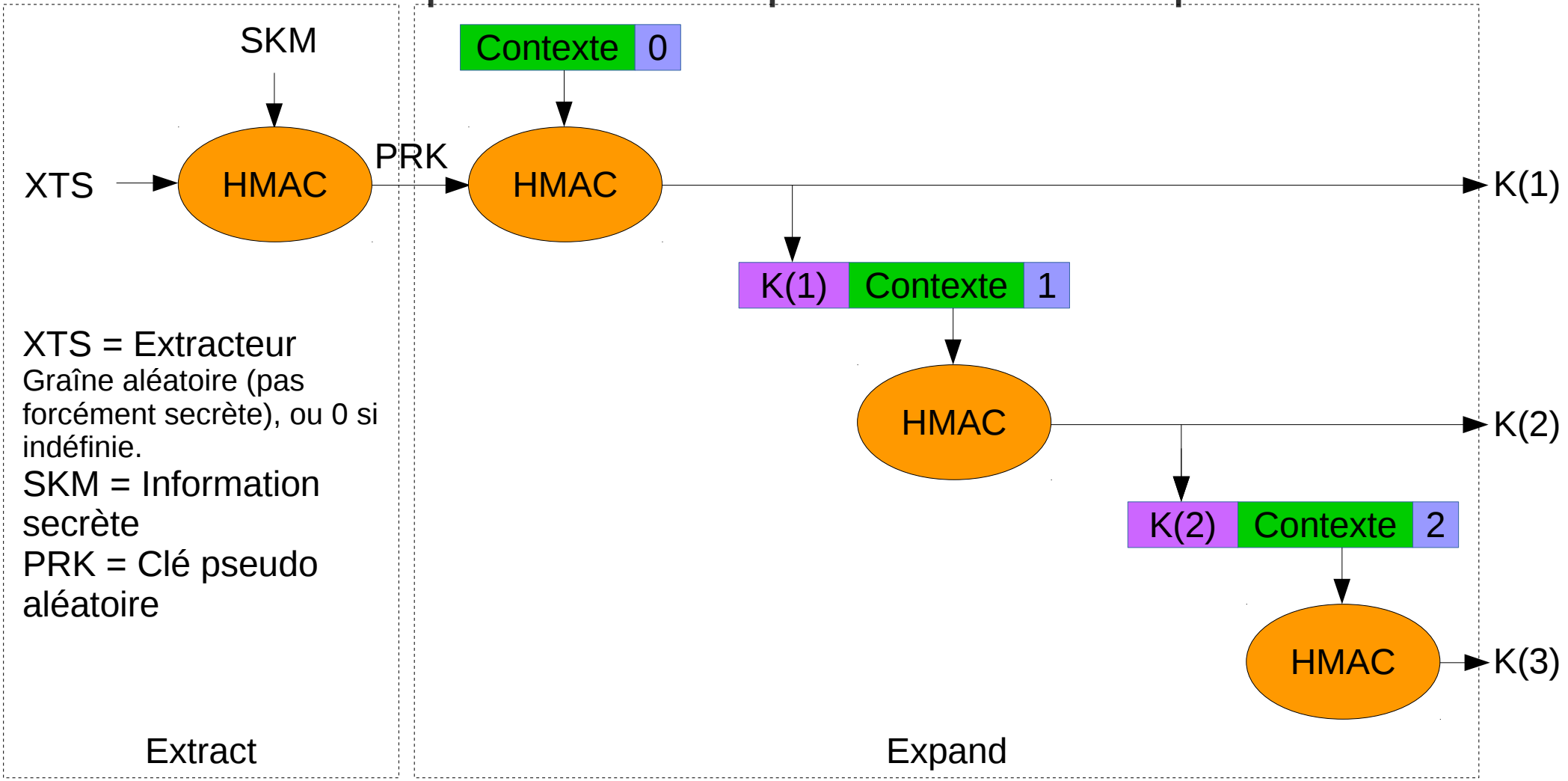
Les algorithmes de dérivation de clé (KDF) en deux étapes

Cas pratique : On utilise SHA-256 pour le HMAC afin de générer une clé pour un chiffrement AES-128 sur 128 bits, et une clé pour un HMAC-SHA1 sur 160 bits.



Les algorithmes de dérivation de clé (KDF) en deux étapes

- Rien n'empêche de remplacer le HMAC par un AES-CBC



Les algorithmes de dérivation de clé (KDF) en deux étapes

- Méthode appelée *Extract-then-Expand*