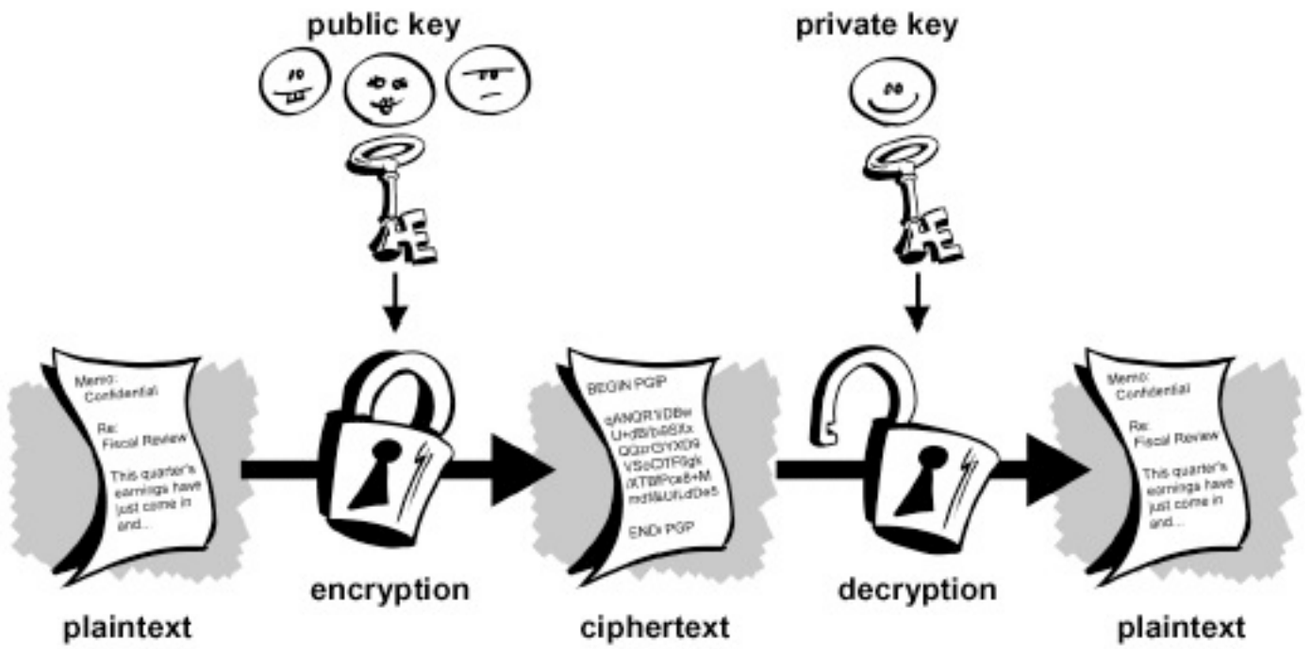
Cryptographie à clé publique

Pierre-Alain Fouque

Diffie-Hellman



- Papier historique «New Directions in Cryptography»
- Opération difficile: inversion du chiffrement (fonction à sens unique à trappe)
- Résolvent le problème de l'échange de clé

Problème Diffie-Hellman

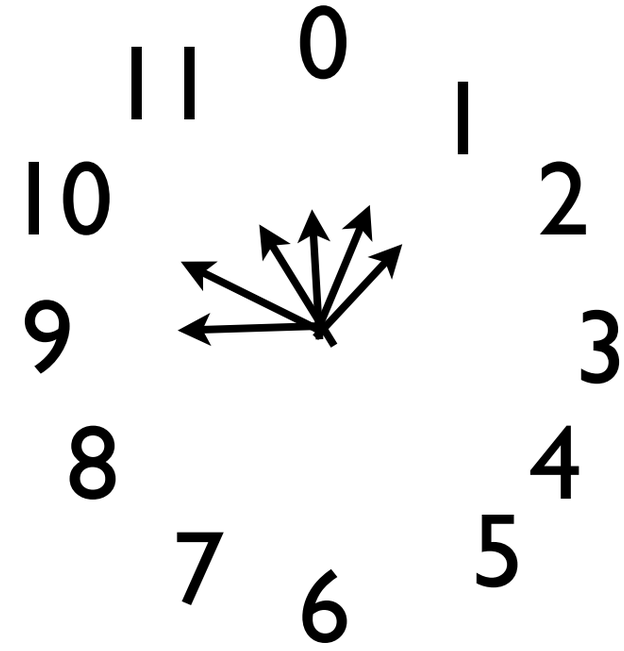
- Soit G un groupe cyclique: tous les éléments s'écrivent de la forme g^x pour x=0 à n-1
- Problème du logarithme discret (DLog): étant donné y, trouver x tel que $y=g^x$?
- Meilleur algorithme: complexité $O(n^{1/2})$
- Problème Diffie-Hellman (DH): étant donné g^x et g^y, calculer g^{xy} ?
- Problèmes équivalents sur une courbe elliptique

Rivest-Shamir-Adleman

- RSA: premier système de chiffrement à clé publique
- résout aussi le problème de la signature numérique (RSA=fonction à sens unique est une permutation)
- Sécurité = un problème algorithmique difficile
  - factorisation des nombres de la forme produit de 2 grands nombres premiers
  - inverser la fonction $f(x)=x^e \bmod N$

Calcul modulaire

- «Groupe de l'horloge»
- $Z_{12}=\{0,1,2,3,...,11\}$
- 2+5=7
- 9+5=14=12+2=2 mod 12



Addition dans Z_7

- $Z_7=\{0,1,2,3,4,5,6\}$
- 2+3=5
- 4+5=9=7+2=2 mod 7

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

Soustraction dans Z_7

- 5-2=3
- 2-5=-3=(-3)+7=4 mod 7

-	0	1	2	3	4	5	6
0	0	6	5	4	3	2	1
1	1	0	6	5	4	3	2
2	2	1	0	6	5	4	3
3	3	2	1	0	6	5	4
4	4	3	2	1	0	6	5
5	5	4	3	2	1	0	6
6	6	5	4	3	2	1	0

Multiplication dans Z_7

- $a \times b = \underbrace{a+a+\ldots+a}_{b \text{ fois}} = \underbrace{b+b+\ldots+b}_{a \text{ fois}}$

- $3 \times 5 = 5+5+5 = 10+5 = 3+5 = 8 = 7+1 = 1 \mod 7$
- $3 \times 5 = 15 = (2 \times 7)+1 = 1 \mod 7$
- $6 \times 4 = 24 = (3 \times 7)+3 = 3 \mod 7$

Multiplication dans Z_7

- $3 \times 5 = (2 \times 7) + 1 = 1 \mod 7$
- $6 \times 4 = (3 \times 7) + 3 = 3 \mod 7$

x	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Multiplication dans Z_6

x	0	1	2	3	4	5	
0	0	0	0	0	0	0	
1	0	1	2	3	4	5	
2	0	2	4	0	2	4	
3	0	3	0	3	0	3	
4	0	4	2	0	4	2	
5	0	5	4	3	2	1	

x	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Multiplication dans Z_6

x	0	1	2	3	4	5	
0	0	0	0	0	0	0	
1	0	1	2	3	4	5	
2	0	2	4	0	2	4	
3	0	3	0	3	0	3	
4	0	4	2	0	4	2	
5	0	5	4	3	2	1	

x	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Exponentiation modulaire

- $a^b = \underbrace{a \times a \times \ldots \times a}_{b \text{ fois}}$
- $4^5 = 4 \times 4 \times 4 \times 4 \times 4 = 16 \times 16 \times 4 = 2 \times 2 \times 4 = 2 \bmod 7$
- $5^6 = 5 \times 5 \times 5 \times 5 \times 5 \times 5 = 4 \times 4 \times 4 = 16 \times 4 = 2 \times 4 = 8 = 1 \bmod 7$

Inverse modulaire

- Inverse de a mod N (noté «*1/a mod N*» ou «a^{-1} *mod N*») = b tel que $a \times b = 1 \bmod N$
- $3^{-1}=5 \bmod 7$
- $5^{-1}=3 \bmod 7$
- $6^{-1}=6 \bmod 7$
- $2^{-1}=4 \bmod 7$

x	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Inverse modulaire

- $5^{-1}=5 \bmod 6$
- $2^{-1}=??? \bmod 6 \Rightarrow$ non défini
- Seuls 1 et 5 sont inversibles modulo 6

x	0	1	2	3	4	5	
0	0	0	0	0	0	0	
1	0	1	2	3	4	5	
2	0	2	4	0	2	4	
3	0	3	0	3	0	3	
4	0	4	2	0	4	2	
5	0	5	4	3	2	1	

Division modulaire

- Diviser *a* par *b* revient à multiplier *a* par l'inverse de *b*: $a/b=a\times b^{-1} \bmod n$
- $2/3=2\times 3^{-1}=2\times 5=3 \bmod 7$ ($3\times 3=2$ *mod 7*)
- 5 / 4 mod 6 n'est pas défini car 4 n'est pas inversible

En résumé

- Z_N={0,1,2,...,N-1} (groupe d'horloge)
- Opérations d'addition, de soustraction, de multiplication
- Opération d'exponentiation
- Parfois, opération d'inversion
  - cas particulier fondamental: si p est «premier», Z_p*={1,2,...,p-1} est constitué d'éléments tous inversibles

Arithmétique modulaire

- But: effectuer tous les calculs modulo un entier *N*
- Règle: «tout entier *x* est remplacé par le reste de la division entière de *x* par *N*»
- Définition de l'anneau Z_N:
  - éléments: entiers de *0* à *N-1*
  - Opérations: addition, soustraction, multiplication, exponentiation, et ... parfois, inversion

Exponentiation

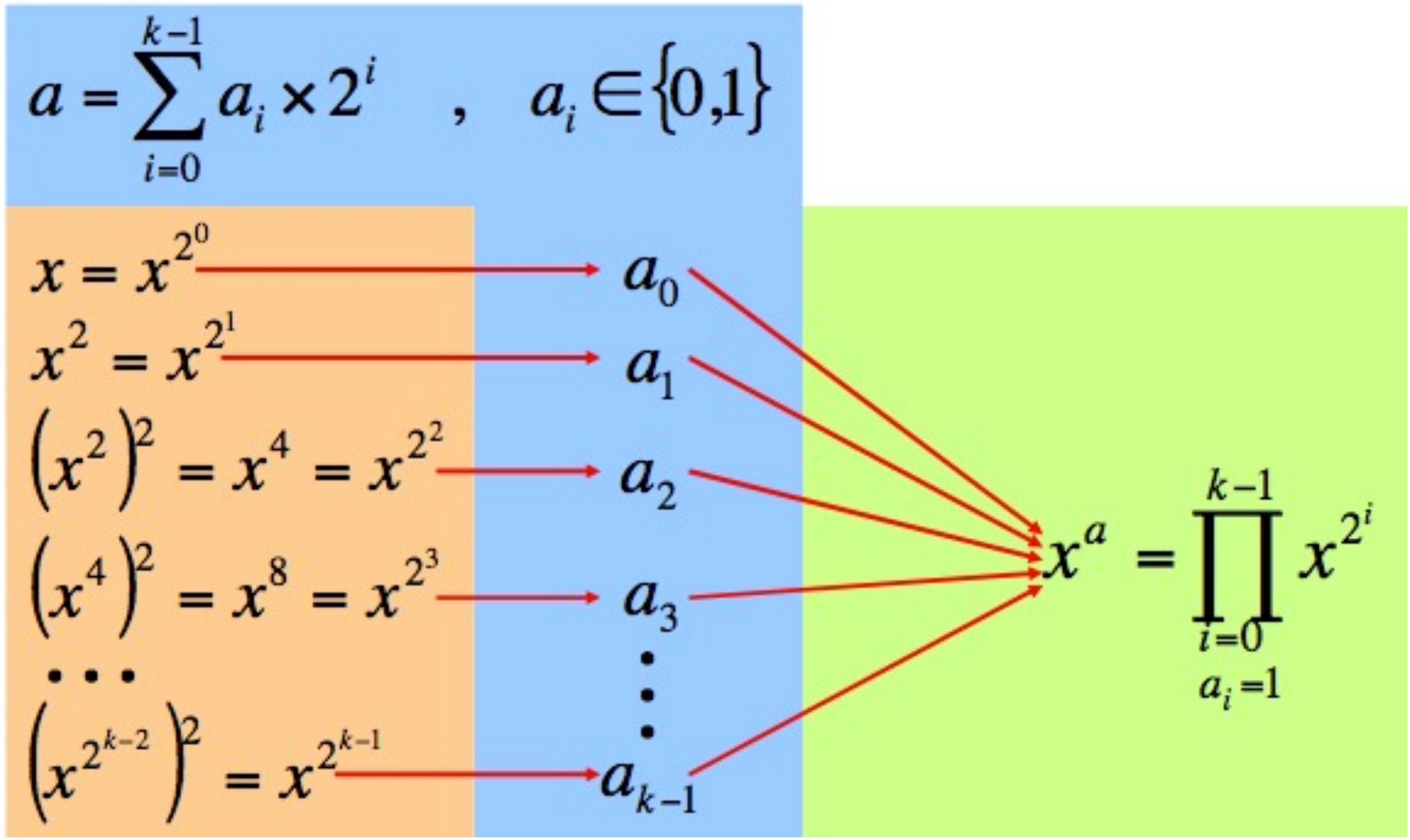
- But: calculer $x^a = x.x.x...x$ (a fois)
- Algorithme élémentaire: a-1 multiplications
- ... mais on peut faire mieux $x^{16} = (((x^2)^2)^2)^2$
- soit 4 multiplications (carrés) au lieu de 15

Exponentiation binaire

$$a=\sum_{i=0}^{k-1} a_i \times 2^i,\ a_i \in \{0,1\}$$

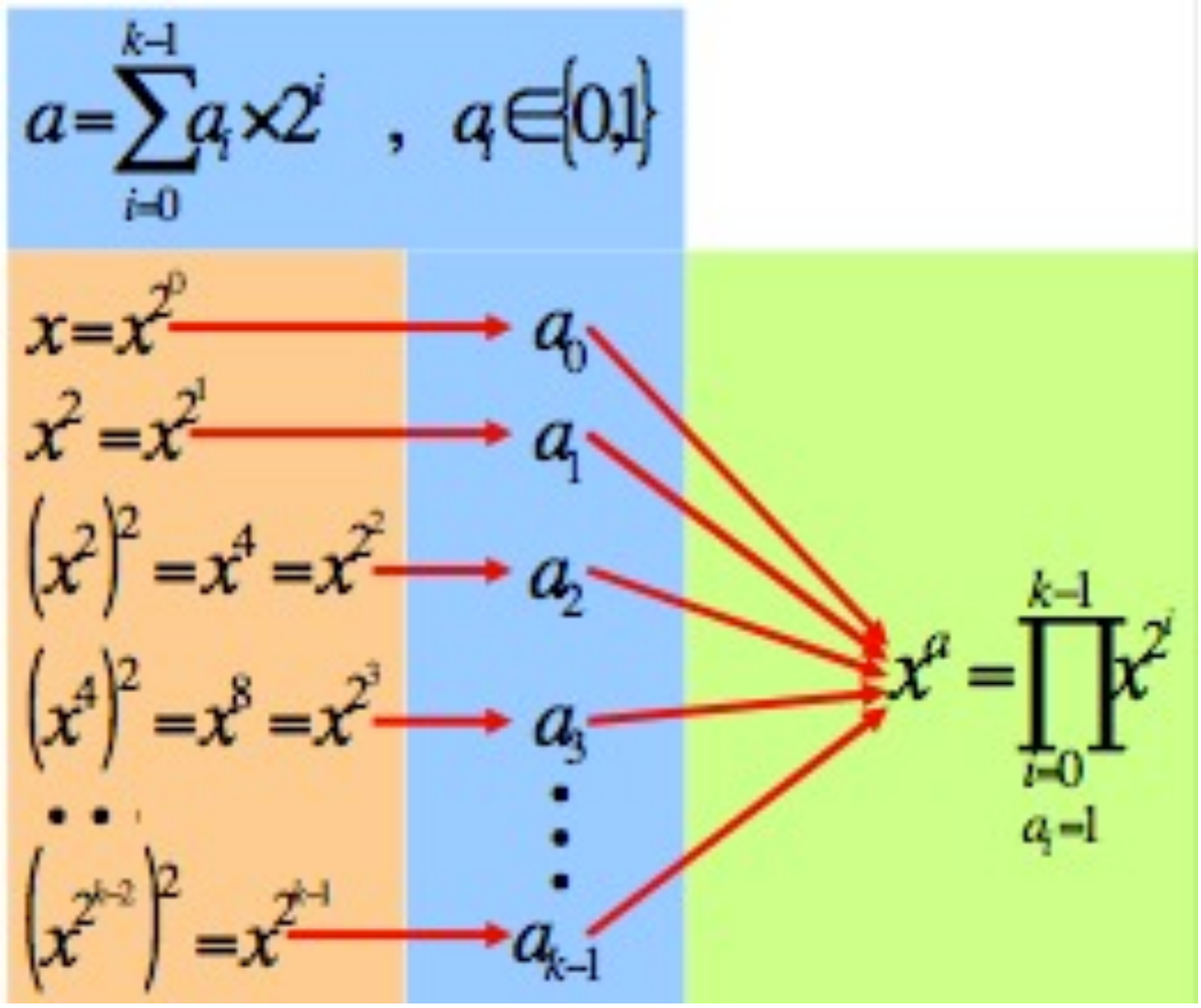
$$x^a = x^{\sum_{i=0}^{k-1} a_i \times 2^i} = \prod_{i=0}^{k-1} x^{a_i \times 2^i} = \prod_{i=0}^{k-1} \left(x^{2^i}\right)^{a_i} = \prod_{\substack{i=0 \\ a_i=1}}^{k-1} x^{2^i}$$

Exponentiation binaire



Exponentiation binaire

- But: calculer x^a
- $y=1$, $P=x$
- tant que $a!=0$ faire
  - si a est impair, $y=y.P$
  - $a=a/2$ (division entière)
  - $P=P^2$
- retourner y



Exponentiation binaire

- But: calculer x^a
- $y=1$, $P=x$
- tant que $a!=0$ faire
  - si a est impair, $y=y.P$
  - $a=a/2$ (division entière)
  - $P=P^2$
- retourner y

y	a	P
1	13	x
x	13	x
x	6	x
x	6	x^2
x	6	x^2
x	3	x^2
x	3	x^4
x^5	3	x^4
x^5	1	x^4
x^5	1	x^8
x^{13}	1	x^8
x^{13}	0	x^8
x^{13}	0	x^{16}

Complexité du calcul x^a

- Combien de fois peut-on «diviser a par 2 » ?
  - si $a=2^k$, $k+1$ fois ($=log_2(a)$)
    - en général, de l'ordre de $log_2(a)$
- à chaque étape, on fait 1 ou 2 multiplications (1,5 en moyenne)
- La complexité du calcul est donc de l'ordre de $1,5.log_2(a)$ multiplications

Exponentiation modulaire

- But: calculer x^a mod N
- Ne surtout pas calculer x^a puis réduire modulo N !
- Exemple: $17^{1234567890} = 1 \bmod 16$ calcul immédiat bien que $17^{1234567890}$ soit impossible à calculer...
- Idée: effectuer les réductions modulaires dès que possible

Exponentiation binaire

- But: calculer $x^a \bmod N$
- $y=1$, $P=x \bmod N$
- tant que $a!=0$ faire
  - si a est impair, $y=y.P \bmod N$
  - $a=a/2$ (division entière)
  - $P=P^2 \bmod N$
- retourner y
- *Remarque:* on ne manipule aucun entier $>N^2$

Exponentiation binaire

- Calcul de x^a mod N
  - $|a|$=1024 bits =>1536 multiplication modulaires
  - a=3=> 2 multiplications modulaires
  - a=216+1=> 16 multiplications modulaires
- Complexité d'une multiplication mod N
  - => environ $|N|^2$ opérations bit à bit
- Exponentiation modulaire: environ $|N|^3$ opérations (taille module x2 => temps de calcul x8)

PGCD

- Plus Grand Commun Diviseur
- PGCD(30,42)=PGCD(5×6,6×7)=6
- PGCD(11,17)=1
- PGCD(22,35)=PGCD(11×2,7×5)=1
- Algorithme (récursif):
  - PGCD(a,0)=a pour tout a
  - PGCD(a,b)=PGCD(b,a mod b)

Algorithme d'Euclide

- Entrée: a et b
- Sortie: d tel que d=PGCD(a,b)
- x=a, y=b
- Tant que y>0
  - r=x-qy (division euclidienne de x par y)
  - x=y
  - y=r
- return x

Inversion mod *N*

- X est inversible modulo *N* ssi il existe Y tel que X.Y=1 mod *N*
- Exemple: 8.7=1 mod *55*
- Seuls les entiers premiers avec *N* sont inversibles (Bézout)
- Calcul d'inverse: Euclide étendu appliqué à X et N: X.u+*N*.v=1 donc X.u=1 mod *N*
  X^{-1}=u mod *N*

Algorithme d'Euclide étendu

- Entrée: a et b d'au plus l-bit
- Sortie: d=pgcd(a,b),u,v tels que d=a.u+b.v
- **x**=(a,1,0), **y**=(b,0,1)
- Tant que y_1>0 faire
  - **r**=**x**-q**y** (q=quotient euclidienne x_1 par y_1)
  - **x**=**y** puis **y**=**r**

Calcul du pgcd si on ne regarde que x1 et y1

Complexité ($O(l^3)$ $O(l^2)$)

- à tout instant: $x_1=ax_2+bx_3$ et $y_1=ay_2+by_3$
- $|y_1|$ décroît (y_1 reste de la division de x_1 par y_1)
- $y_1=0$ (donc le programme s'arrête)

- $pgcd(x_1,y_1)=pgcd(y_1,x_1-qy_1)$
- $O(l^2)$ division euclidienne et $O(l)$ itérations
- $(z_0,z_1,...,z_i)$ valeurs prises par y_1 (suite décroissante)
- $z_0<2^l$, $z_i=0$ et $z_{j+1}=z_{j-1}-q_jz_j$ avec $q_j=z_{j-1}/z_j$
- $z_{j+1}+z_j \leq z_{j-1}$ car $q_j>0$ (sauf début) puis Fibonacci

Calcul d'inverse modulaire

- Calculer a^{-1} mod b: au+bv=1 ⇒au=1 mod b

- Inverse de 22 mod 3 (a=22, b=35)

itération	x	y	q
0	(22,1,0)	(35,0,1)	0
1	(35,0,1)	(22,1,0)	1
2	(22,1,0)	(13,-1,1)	1
3	(13,-1,1)	(9,2,-1)	1
4	(9,2,-1)	(4,-3,2)	2
5	(4,-3,2)	(1,8,-5)	4
6	(1,8,-5)	(0,-35,22)	

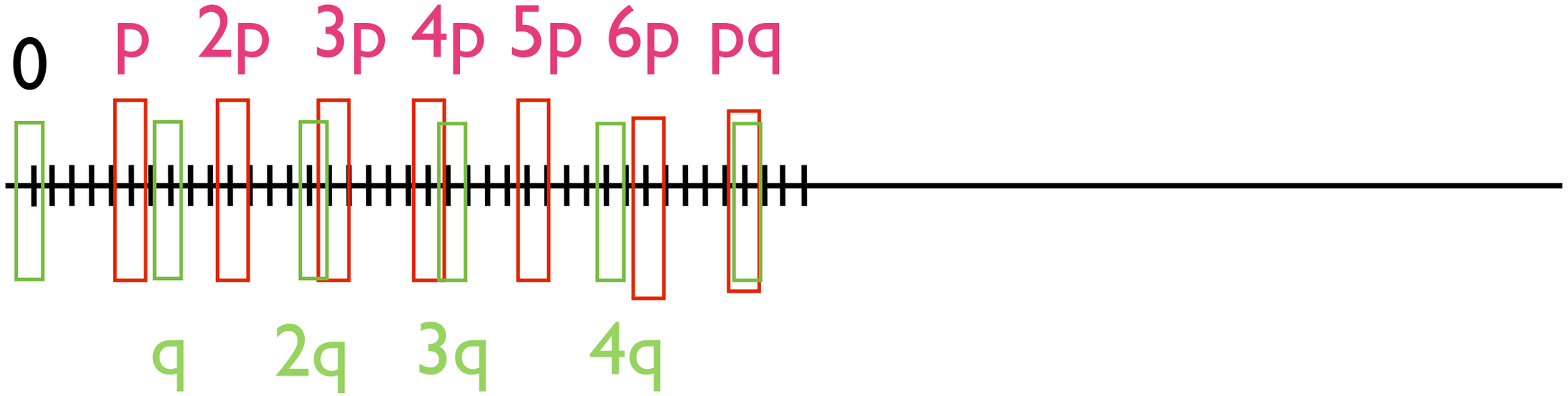
1=22x8-5x35

Z_N^*

- On note Z_N^* l'ensemble des entiers de Z_N inversibles modulo N
- Si N est premier, $Z_N^*=\{1,2,...,N-1\}$
- Z_N^* = entiers de 1 à $N-1$ premiers avec N
- Indicatrice d'Euler: $Phi(N)=Card(Z_N^*)$ si $N=p_1^{e1}.p_2^{e2}....p_k^{ek}$,
  $Phi(N)=p_1^{e1-1}(p_1-1)....p_k^{ek-1}(p_k-1)$

Calcul de Phi(*N*)

- Z_N^* entiers de 1 à *N-1* *premiers avec* *N*



- Phi(p.q)=pq-(q-1)-(p-1)-1=(p-1).(q-1)

Théorème d'Euler

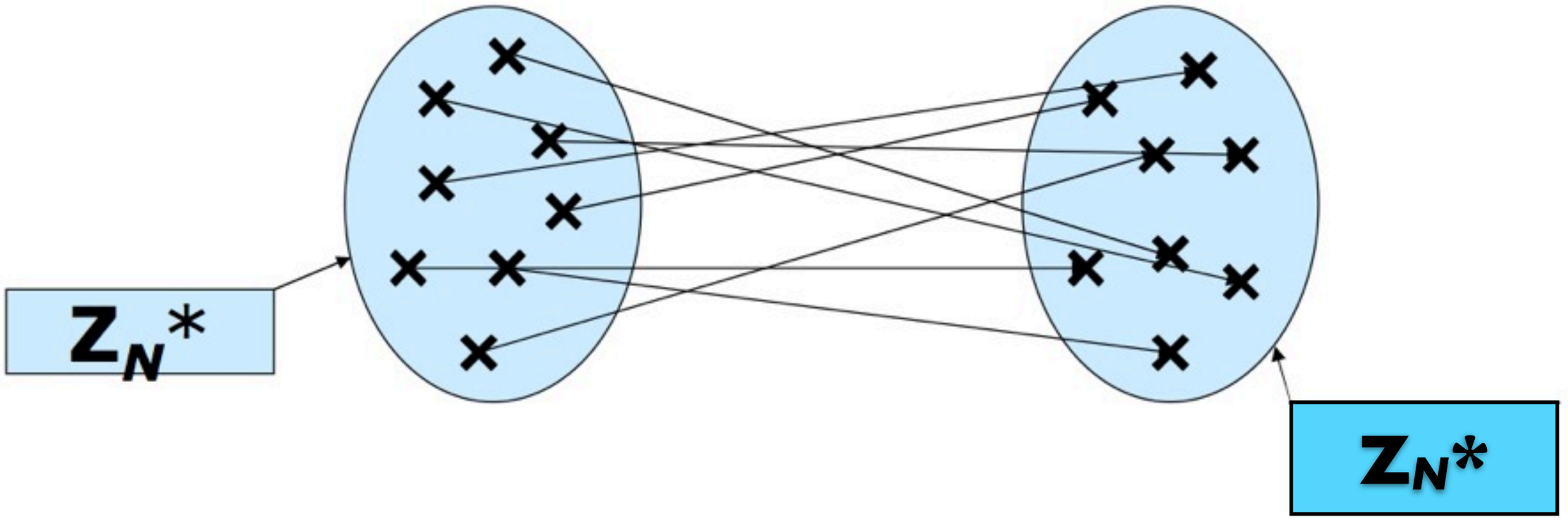
$g \in \mathbf{Z}_N^*$

$\mathbf{Z}_N^* \longrightarrow \mathbf{Z}_N^*$

$x \longmapsto g.x$

$y/g \longleftarrow\!\shortmid\ y$

$\mathbf{Z}_N^* = \{x_1, x_2, \ldots x_\varphi\}$ $\quad g.x_i = x_{\sigma(i)}$



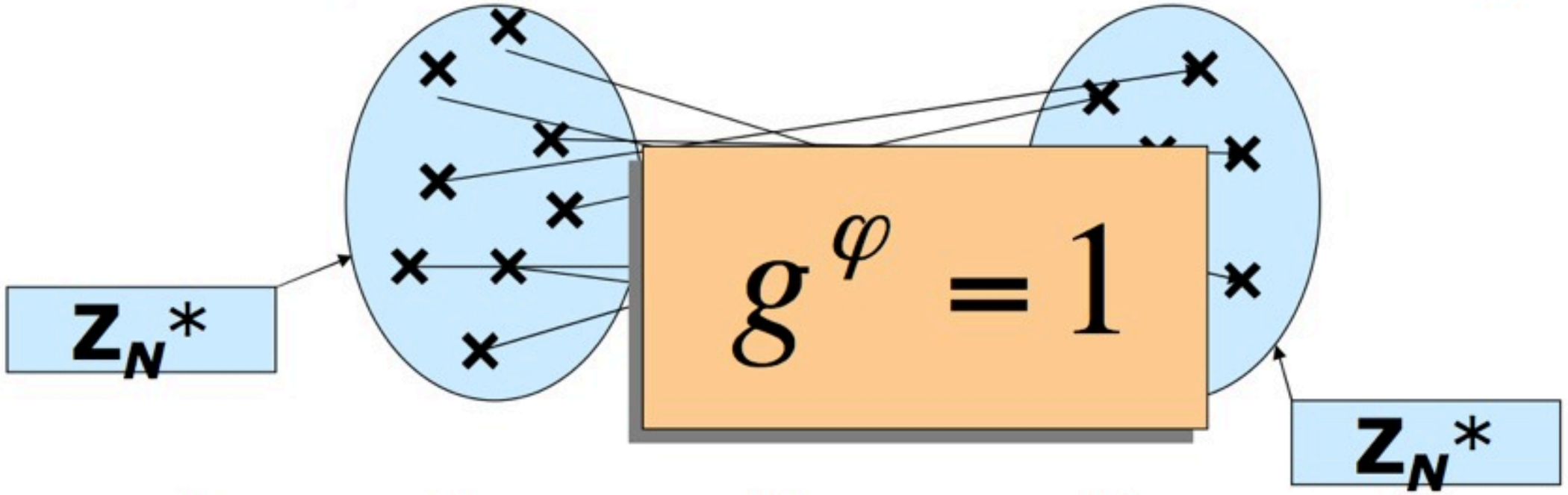
Théorème d'Euler

$g \in \mathbf{Z}_N^*$

$\mathbf{Z}_N^* = \{x_1, x_2, \ldots x_\varphi\}$

$\mathbf{Z}_N^* \longrightarrow \mathbf{Z}_N^*$

$x_i \longmapsto g.x = x_{\sigma(i)}$



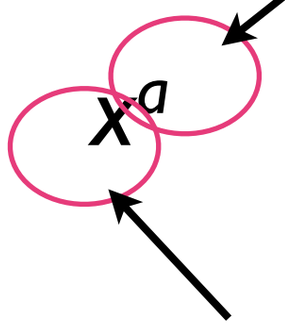
$$g^{\varphi} \times \prod_{i=1}^{\varphi} x_i = \prod_{i=1}^{\varphi} g\,x_i = \prod_{i=1}^{\varphi} x_{\sigma(i)} = \prod_{i=1}^{\varphi} x_i$$

Théorème d'Euler

Quelque soit $x \in Z_N^*$, $g^o = 1 \mod N$, $o = \text{Card}(Z_N^*)$

Quelque soit k, $g^{a+ko} = g^a \mod N$

Les exposants vivent
modulo phi(N)



Les éléments vivent
modulo N

Cas p premier

- Petit théorème de Fermat: Si $N=p$ premier, on a $x^{p-1}=1 \mod p$ pour tout $x!=0 \mod p$
- $Zp^*=\{1,2,...,p-1\}$ et phi$(p)=p-1$
- *Zp^* est un groupe cyclique $=\{1,g,g^2,...,g^{p-2}\}$ avec phi$(p-1)$ générateurs g*
- *$ord(x^a \mod p)=ord(x)/pgcd(a,p-1)$*

Anneau et corps

- Si n n'est pas premier, on note Zn l'ensemble des entiers entre 0 et n-1, muni des opérations d'addition modulo n et de multiplication mod n, c'est un anneau
  - Autre anneau: Z
- Si n=p premier, alors Zp est un corps: c'est-à-dire que c'est un anneau et pour tout élément non nul, il existe un inverse modulaire
  - Autres corps: Q, R, C

Théorème des restes chinois

- Th: Soit m,n deux entiers tels que pgcd(m,n)=1
- f:(Zmn)→(Zm)x(Zn) f(x)=(x mod m, x mod n) est un isomorphisme d'anneau
- phi(mn)=phi(m)xphi(n)
- $f^{-1}(a,b)=an(n^{-1} \bmod m) + bm(m^{-1} \bmod n) \bmod mn$

Exemple

- Résoudre dans Z_{35}: x mod 5=3 et x mod 7=4
- $f^{-1}(3,4)=(3x7x(7^{-1} \bmod 5)+4x5x(5^{-1} \bmod 7)) \bmod 35$
- $1=(-2)x7+3x5$, donc $5^{-1} \bmod 7=3$ et $7^{-1} \bmod 5=3$
- $f^{-1}(3,4)=(3x7x3+4x5x3) \bmod 35=123 \bmod 35=18$
- Application: calculer rapidement $2^{23} \bmod 35$
  - $2^{23} \bmod 5=2^3 \bmod 5=3$
  - $2^{23} \bmod 7=2^5 \bmod 7=4$
  - $f^{-1}(3,4)=18$

Résidus quadratiques

- Déf: x est un résidu quadratique mod N est un carré modulo N ssi il existe y tel que $x=y^2 \bmod N$
- Soit p un nombre premier impair
  - $x \in Zp^*$ est un résidu quadratique ssi $x^{p-1/2}=1 \bmod p$
  - il y a (p-1)/2 résidus quadratiques dans Zp*
  - si p=3 mod 4, tout résidu quadratique x a deux racines carrées $x^{(p+1)/4} \bmod p$ et $-x^{(p+1)/4}$

Génération de nombres premiers

- Problème de primalité: savoir si un entier x est un nombre premier ou non
- Ce problème a été prouvé dans la classe de complexité P en 2003 (Agrawal, Kayal, Saxena)
- Construire un test de primalité plus efficace

Algorithme de division par essai (Trial division)

- Entrée: un entier n
- Sortie: listes des facteurs premiers de n
- Complexité: $\sqrt{n}$ opérations arithmétiques

  - $b=\sqrt{n}$, x=n, i=2
  - TQ (x>1 et $i \leq b$) faire:
    - TQ (i divise x) affiche i, x=x/i, $b=\sqrt{x}$
    - i=i+1
    - si x>1, affiche x

Tests de primalités

- Test de Fermat: pour tout b>0, $b^{p-1}=1 \mod p$
  - Problèmes: complexité grande et il existe des nombres de Carmichael, non premier, qui vérifie ce test
- Test de Solovay-Strassen: pour tout b>0, $b^{p-1/2}$=symbole Legendre de x par rapport à p
  - Très bien, mais moins efficace que Miller-Rabin

Test de Miller-Rabin

- Entrée: N entier de n bits et k un entier
- Sortie: pseudo-primalité de N
- Complexité: $O(kn^3)$
  - Si N=2, return «premier»
  - Si N pair, return «composé»
  - $N=2^s t+1$ avec t impair
  - Pendant k itérations, faire
    - Prendre aléa $0<b<n$, $x=b^t \bmod N$, i=0
    - Si x!=1, TQ x!=N-1, $x=x^2 \bmod N$, i=i+1
    - Si i=0 ou x=1, return «composé» FinPendant
    - return «pseudo-premier»

Analyse Miller-Rabin

- Théorème: Soit N un entier, et considérons l'algo MR avec comme paramètre N et k.
  - Si N est premier, l'algorithme retourne «premier» ou «pseudo-premier».
  - Réciproquement, si N est composé, l'algorithme retourne «composé» avec proba sup. à $1-4^{-k}$.
- Si N est premier, $b^{N-1}=1 \bmod N$ pour tout b. Comme Zp est un corps, 1 a 2 racines carrées 1 et -1 mod N. Soit $b^t=1 \bmod N$, soit il existe $0 \leq i < s$, tel que $b^{2it}=1 \bmod N$.
- Si N est composé, il a au mois 4 racines carrées, dont 2 révèlent la factorisation de N.
  - $1=(1 \bmod p, 1 \bmod q)$ et $-1=(-1 \bmod p, -1 \bmod q)$
  - $x=(1 \bmod p, -1 \bmod q)$ et $-x=(-1 \bmod p, 1 \bmod q)$ sont tq $\text{pgcd}(x+1,N)=q$

Génération de nombres premiers

- Théorème des Nombres Premiers: Soit pi(N) le nombre de nombres premiers dans {2,3,...,N}. $p(N) \simeq N/\ln N$
  - Le nombre de nombres premiers $\leq 2^n$ est $2^n/n$, soit environ $1/n$.
- Au bout de k essais aléatoires, on a une probabilité petite de ne pas tomber sur un nombre premier $\leq (1-1/n)^k$

Algorithmes de factorisation

- Problème de la factorisation: étant donné un entier N, trouver des facteurs non-triviaux (différents de 1 et N) de N
  - Plusieurs algorithmes dont la complexité est exponentielle en n (n=log(N)).
  - Meilleur algorithme (NFS) a une complexité en $c(N)=\exp(O(\log(N)^{1/3}\ \mathrm{loglog}(N)^{2/3}))$
  - Pour n=1024, $c(n)=2^{80}$

RSA

- Génération des clés: publique (e,N), privée (d,N)
  - Générer 2 nombres premiers p et q aléatoirement de 1024 bits
  - Calculer N=pq, puis phi(N)=(p-1)(q-1)
    - Prendre e tel que pgcd(e, phi(N))=1 et inverser e mod phi(N)=d
- Chiffrement: $f(x)=x^e \bmod N$
- Déchiffrement: $f^{-1}(y)=y^d \bmod N$

RSA-CRT

- Génération des clés: publique (e,N), privée (dp,dq,p,q)
  - Générer 2 nombres premiers p et q aléatoirement de 1024 bits
  - Calculer N=pq, puis phi(N)=(p-1)(q-1)
  - Prendre e tel que pgcd(e, phi(N))=1 et inverser e mod phi(N)=d
  - dp=d mod (p-1)
  - dq=d mod (q-1)

RSA-CRT

- Chiffrement: $f(x)=x^e \bmod N$
- Déchiffrement: $f^{-1}(y)$ clé privée (dp,dq,p,q)
  - $c_1=(y \bmod p)^{dp} \bmod p$
  - $c_2=(y \bmod q)^{dq} \bmod q$
  - Combiner c_1 et c_2 avec le théorème des restes chinois: facteur 4 en temps

Sécurité de RSA

- RSADP (Decryption Problem):
  - Entrée: (e,N) clé publique RSA et message chiffré y
  - Sortie: x tel que $y=x^e \bmod N$
- RSAKRP (Key Recovery Problem):
  - Entrée: (e,N) clé RSA
  - Sortie: d tel que $x^{ed} \bmod N=x$ pour tout $x \in Z_N^*$
- RSAEMP (Exponent Multiple Problem):
  - Entrée: Module RSA N
  - Sortie: entier k tel que $x^k \bmod N=1$ pour tout $x \in Z_N^*$

Sécurité de RSA

- RSAOP (Order Problem):
  - Entrée: N module RSA
  - Sortie: phi(N)

- RSAFP (Factorization Problem):
  - Entrée: N module RSA
  - Sortie: p et q tels que N=pq

Sécurité de RSA

- RSADP (Problème de déchiffrement)
- RSAKRP (Problème de retrouver clé secrète)
- RSAEMP (Problème multiple exposant)
- RSAOP (Problème de trouver l'ordre)
- RSAFP (Problème de la factorisation)

$$\begin{array}{ccc} \text{RSAEMP} & \Rightarrow & \text{RSAFP} \\ \Uparrow & & \Updownarrow \\ \text{RSADP} & \Rightarrow \text{RSAKRP} \Leftarrow & \text{RSAOP} \end{array}$$

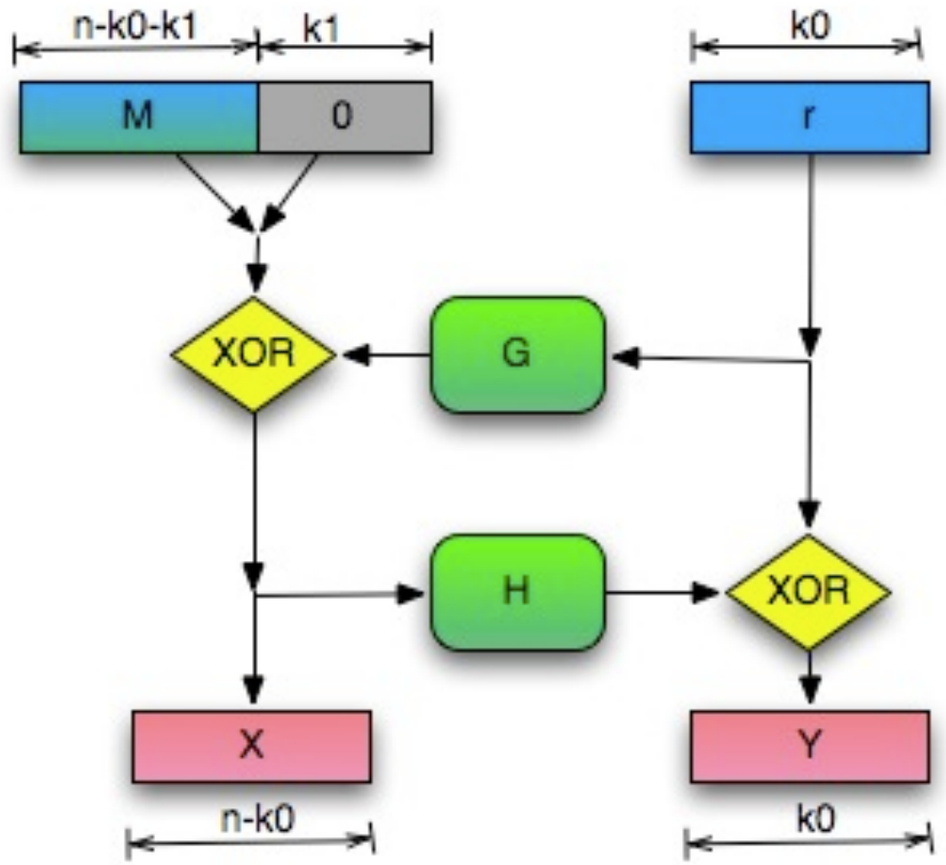
RSA PKCS#1 v1.5

- On peut montrer qu'il est difficile de trouver les bits de poids faible de x connaissant x^e mod N
- Eviter attaque petit exposant
- Attaque par broadcast
- Attaque même module
- Randomisation
- Attaque par canaux auxiliaires
- Sécurité sémantique sous le problème RSA

Sécurité de RSA

- Problème broadcast
- Problème si le message est trop petit $m<N^{1/e}$, c'est facile à inverser
- Chiffrement déterministe
- Attention: D(E(m))=m mod N
- Solution: padding
  - PKCS #1 version 1.5: P(m)=00||01||random||m
  - PKCS #1 version 2: OAEP

RSA-OAEP



- Aujourd'hui, on sait comment utiliser RSA
- Réduction: si on sait attaquer RSA avec une attaque IND-CPA, alors on sait casser plus efficacement le problème RSA

Chiffrement El Gamal

- Clé publique $y=g^x$ et clé secrète x
- Chiffrement: $E(m;r)=(g^r,m.y^r)=(A,B)$
- Déchiffrement: $D(A,B)=B/A^x=m.y^r/g^{rx}=m$
- Sécurité:
  - Un élément au hasard g^r apparaît comme un élément random. La multiplication opère comme le XOR dans le OTP

Signature

Signatures électroniques

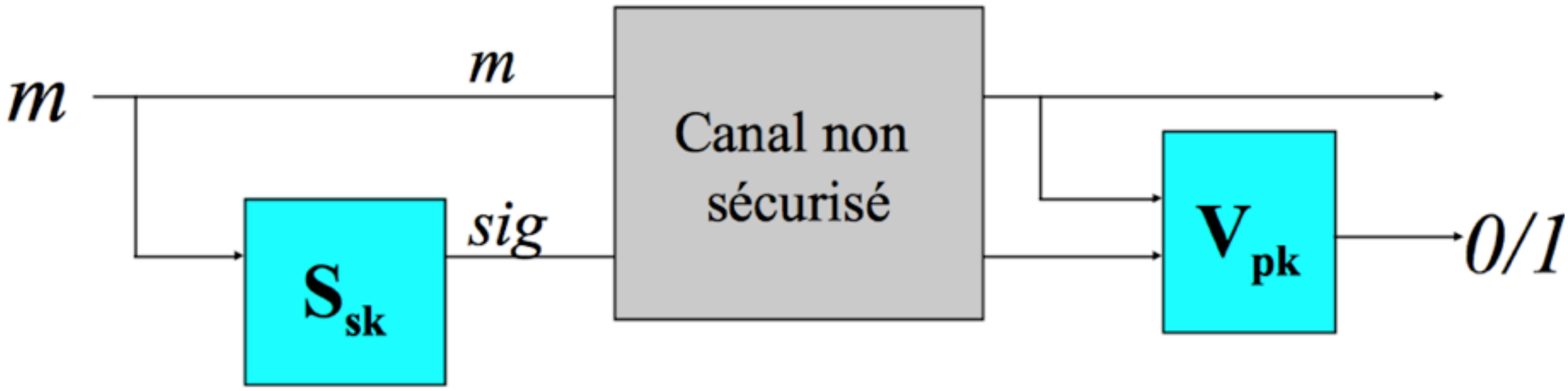
Elles garantissent :

- l'intégrité du document
- l'authentification de l'émetteur
- la non-répudiation

Elles dépendent :

- du message
- du signataire identifié par sa paire de clés publique/secrète

Schéma de signature



Algorithme de génération de signature, S_{sk}

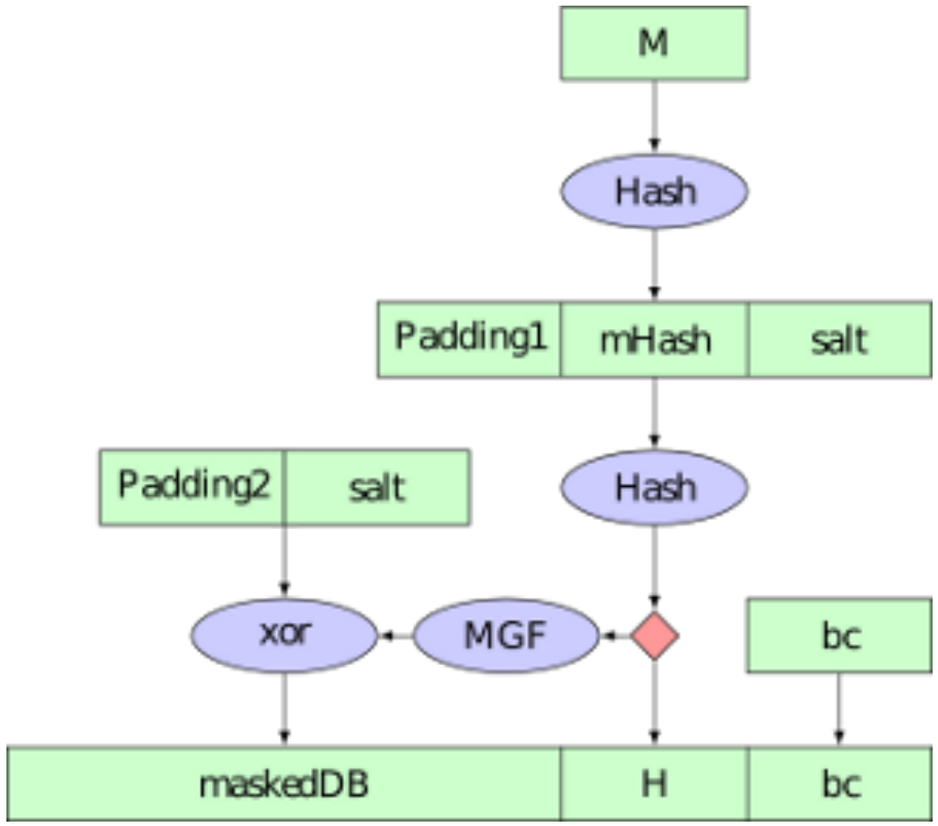
Algorithme de vérification, V_{pk}

Sécurité: impossible de trouver un nouveau couple (m,sig) pour un message m non déjà signé

Signature RSA

- Signature: $S(m)=f(m)^d \bmod N$
- Vérification: $S(m)^e \bmod N = f(m)$?
- Problème: si on a une signature s_1 pour le message m_1 et s_2 pour le message m_2, alors $s_1 x s_2 \bmod N$ est une signature pour le message $m_1 x m_2 \bmod N$
- Signer le haché du message. Collision sur H ?
- f(m) est une fonction de padding:
  - PKCS #1 v1.5 f(m)=00||02||FF...FFFF||H(m)
  - PKCS #1 v2.0 RSA-PSS(m)

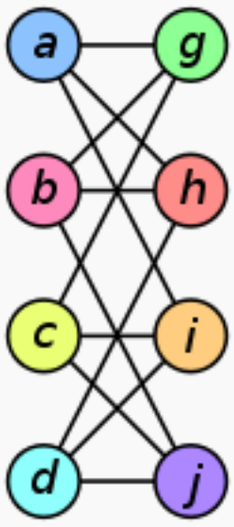
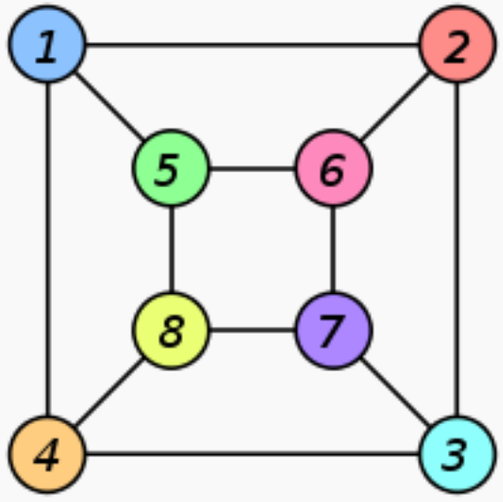
RSA-PSS



- Schéma de signature randomisé
  - si on signe le même message 2 fois, on n'aura pas la même signature
- Hash et MGF sont des fonctions de hachage (concaténé pour la seconde)
- Paddings contient en info sur la fonction de hachage
- salt contient le nombre d'octets de la taille de la fonction de hachage

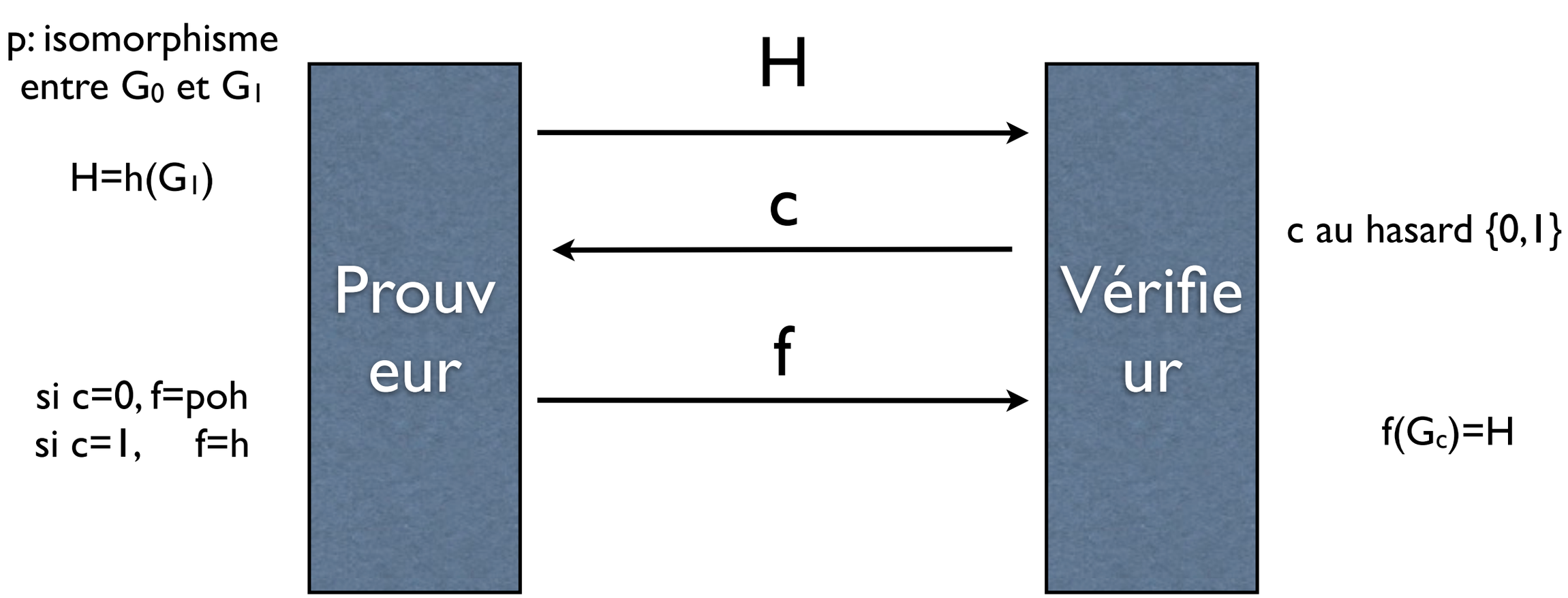
Preuve Zero-Knowledge

- Preuve de connaissance d'un secret
- Après un échange de message, le Vérifieur est certain que le Prouveur connaît un secret avec probabilité $1-2^{80}$
- Mais le Vérifieur n'a aucune information sur le secret détenu par le prouveur
- Problème d'isomorphisme de graphes est conjecturé difficile

Graphe G	Graphe H	Isomorphisme entre G et H
		$f(a) = 1$ $f(b) = 6$ $f(c) = 8$ $f(d) = 3$ $f(g) = 5$ $f(h) = 2$ $f(i) = 4$ $f(j) = 7$

Preuve Zero-Knowledge

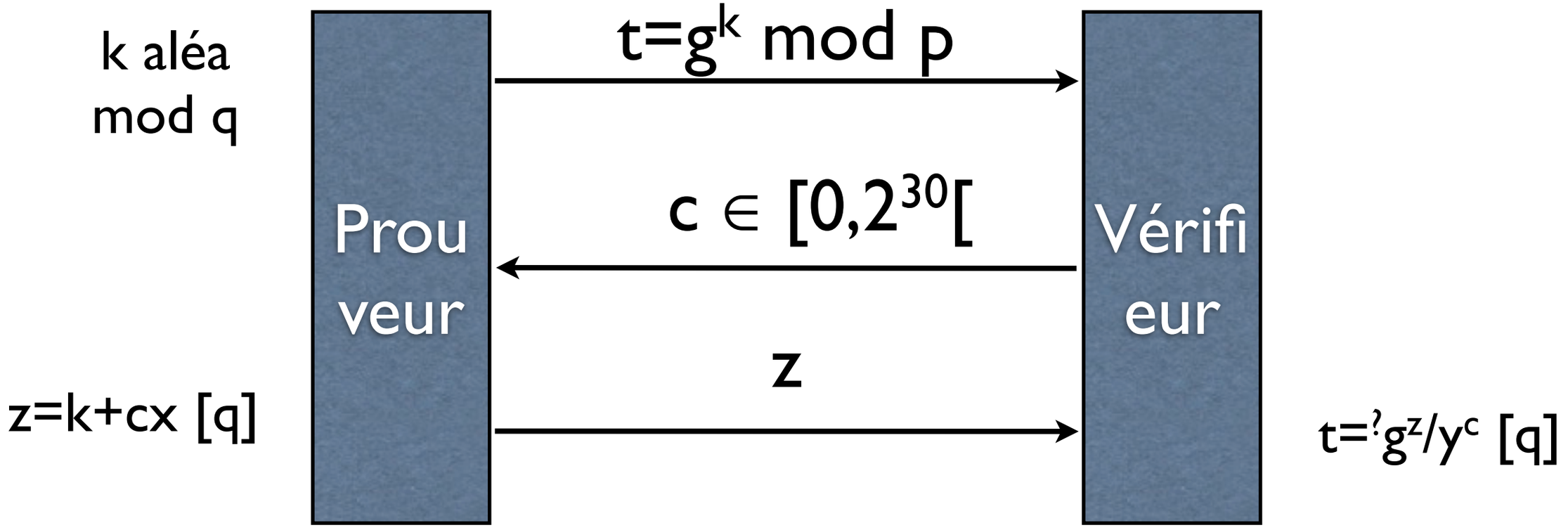
- Le prouveur et le vérifieur connaissent les graphes G_0 et G_1
- Le prouveur connaît un isomorphisme p entre ces graphes
- Le vérifieur avec proba 1/2 attrape le prouveur s'il ment
- Le prouveur qui connaît le secret répond toujours correctement



Vérifieur n'apprend aucune information sur p !

Schéma d'authentification de Schnorr

- Paramètres généraux: (p,q,g) tels que g engendre un sous-groupe de taille q dans le groupe Z_p^*
- Clé publique/secrète d'un utilisateur: (g^x mod p,x)
- Preuve Zero-Knowledge:



Prouveur attrapé avec proba $1-2^{30}$

Signature DSA/ECDSA

- Adaptation de la signature de Schnorr breveté
- Paramètres généraux: (p,q,g) tels que g engendre un sous-groupe de taille q dans le groupe Z_p^*
- générer q de 160 bits, puis p=2qR+1 avec R random de 1024-160 bits
- g aléatoire et calculer $g^{(p-1)/2R} \bmod p \neq 1$
- Clé publique/secrète d'un utilisateur: (p,q,g,$y=g^x \bmod p$) et x<q

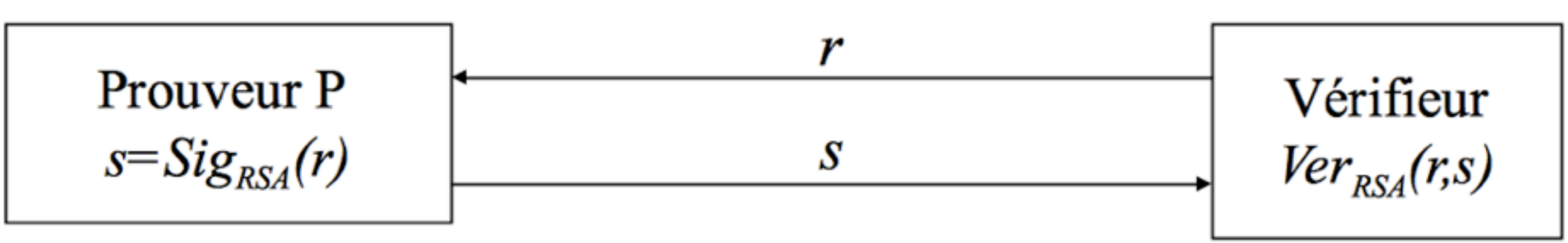
Signature DSA

- Génération Signature
- $s1=(g^k \bmod p) \bmod q$
- $s2=(H(m)+s1*x)/k \bmod q$
- $S(m)=(s1,s2)$

- Vérification Signature
- $0<s1,s2<q$
- $w=s2^{-1} \bmod q$
- $u1=H(m)*w \bmod q$
- $u2=s1*w \bmod q$
- $v=(g^{u1}*y^{u2} \bmod p) \bmod q =? s1$

Identification asymétrique

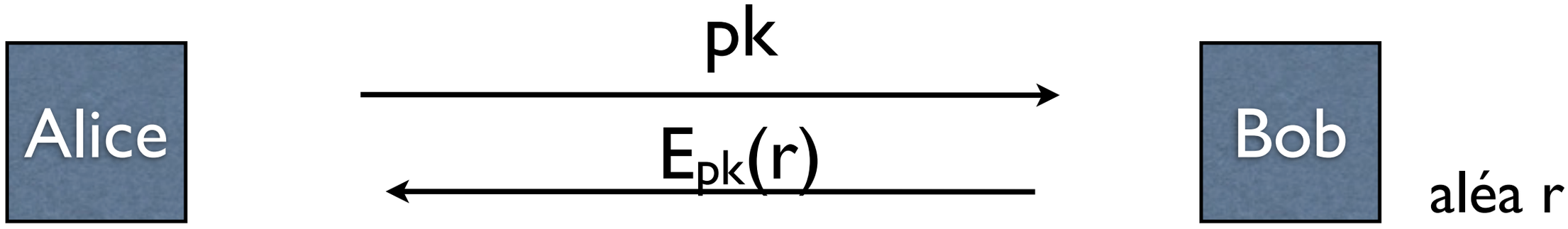
- Protocole Challenge/Response [X.509]
- P possède une clé publique/secrète de signature



Problème: Ne pas utiliser une même clé de signature pour s'identifier et signer. Pourquoi ?

Echange de clés

Faux échange de clé



- Clé commune r
- Bob a le contrôle de la clé
- Avantage: si Alice prouve sa connaissance de sa clé, elle s'authentifie
- utilisé dans SSHv1 et SSL

Diffie-Hellman key exchange

- Alice:
  - génère a au hasard
  - calcule g^a
  - réception de g^b, calcule $K=(g^b)^a=g^{ab}$
- Bob:
  - génère b au hasard
  - calcule g^b
  - réception de g^a, calcule $K=(g^a)^b=g^{ab}$

g^a →

← g^b

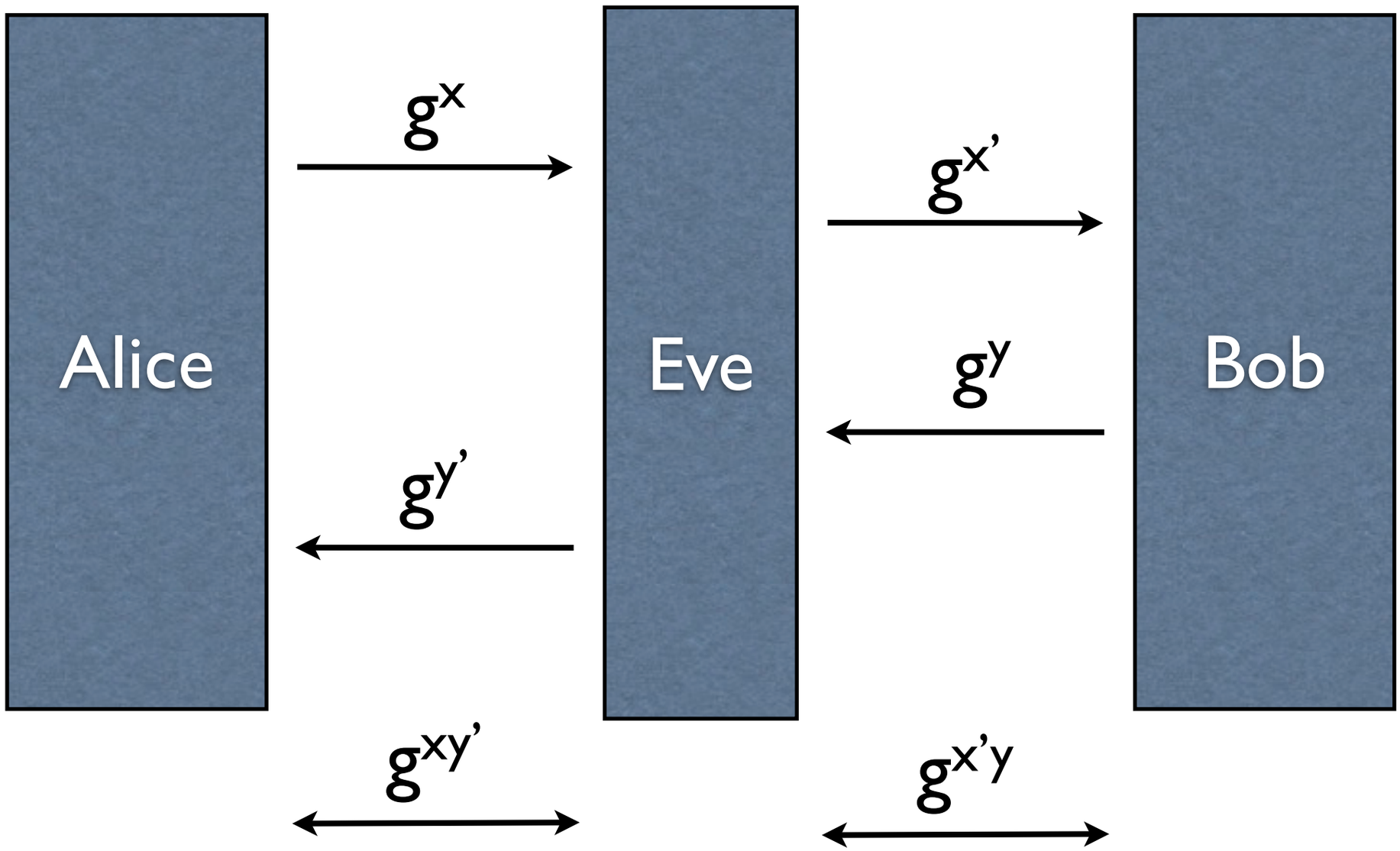
Forward-secure: pas de clé long-terme

Clés éphémères et forward-secrecy

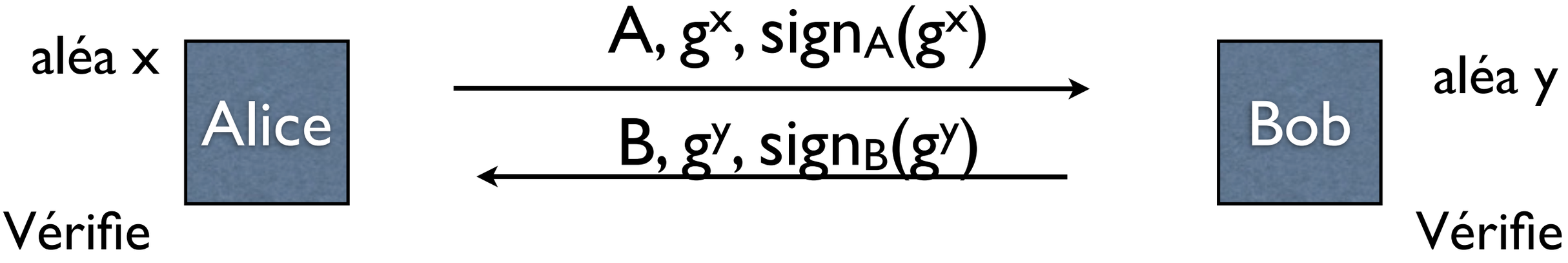
- Forward-secure: les clés de session ne sont pas mises en danger par la fuite des clés long-terme
- Moyen d'y parvenir: utiliser des clés publiques éphémères pour protéger la clé de session
  - éventuellement en combinaison avec des clés long-terme
- Les clés privées éphémères sont effacées sitôt l'échange terminé
- ex: «server key» de SSH v1

Echange de clé: attaque active

Man-in-the-middle Attacks

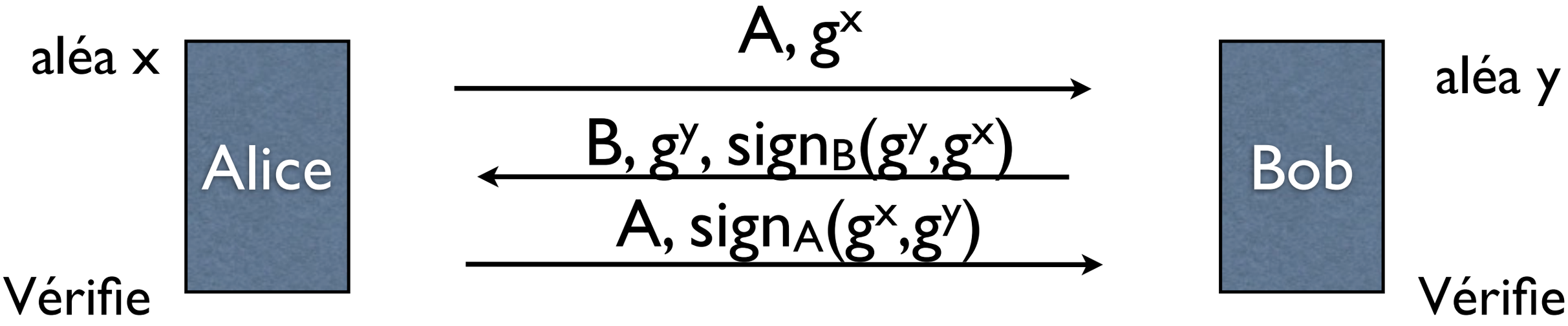


Diffie-Hellman signé v1



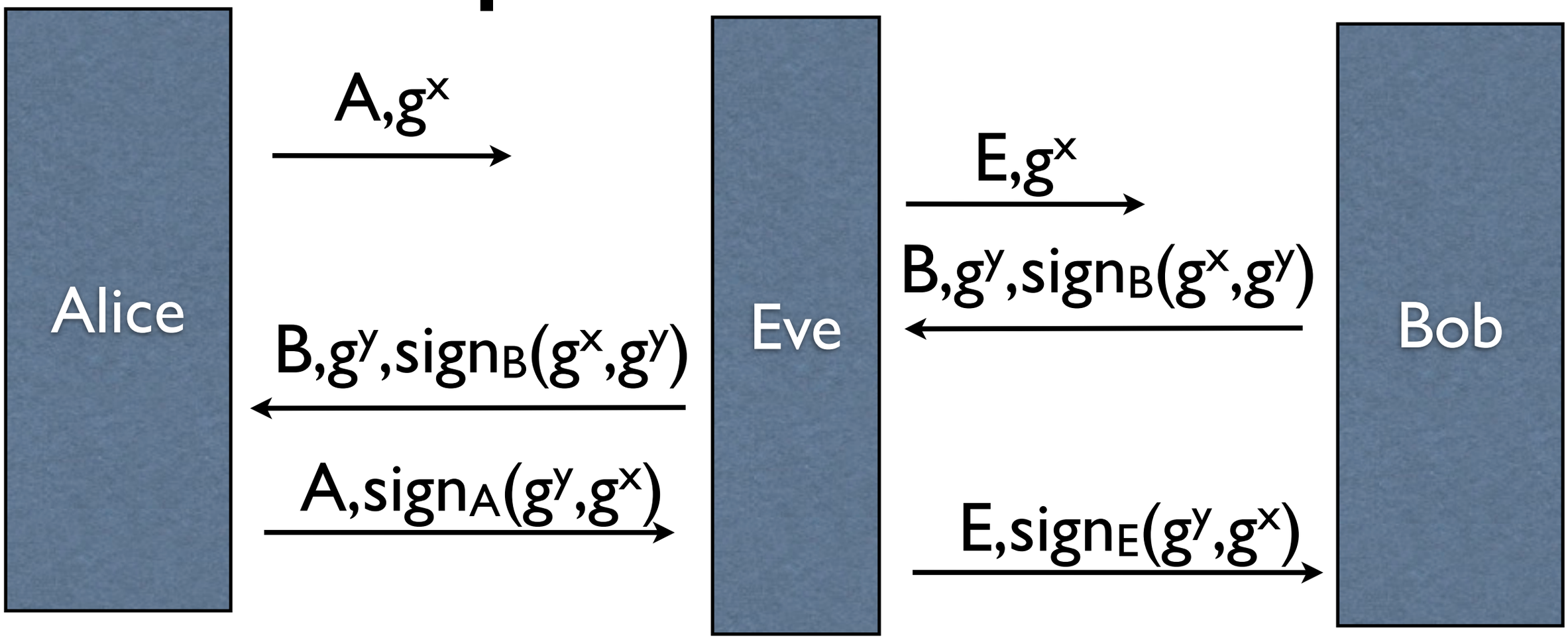
- Evite les attaques par le milieu: impossible pour un attaquant de fournir $sign_A(g^{x'})$
- Rejeu possible: si (g^x,$sign_A(g^x)$) est capturé, il peut être utilisé pour s'authentifier comme A, mais on ne peut pas calculer la clé correspondante

Station-To-Station Protocol



- Le meilleur des deux mondes:
  - évite les attaques par le milieu: impossible pour un attaquant de fournir $sign_B(g^{x'},g^y)$
  - évite le rejeu: la valeur DH est aussi signée
  - Mais...

Usurpation d'identité



- Tous les messages envoyés par Alice sont vus par Bob comme venant d'Eve
- Eve ne connaît pas la clé g^{xy}

Usurpation d'identité

- Bob est une banque
- Alice et Eve des clients
- Alice crédite sont compte avec de la monnaie électronique
- la banque crédite le compte d'Eve

Diffie-Hellman signé v3

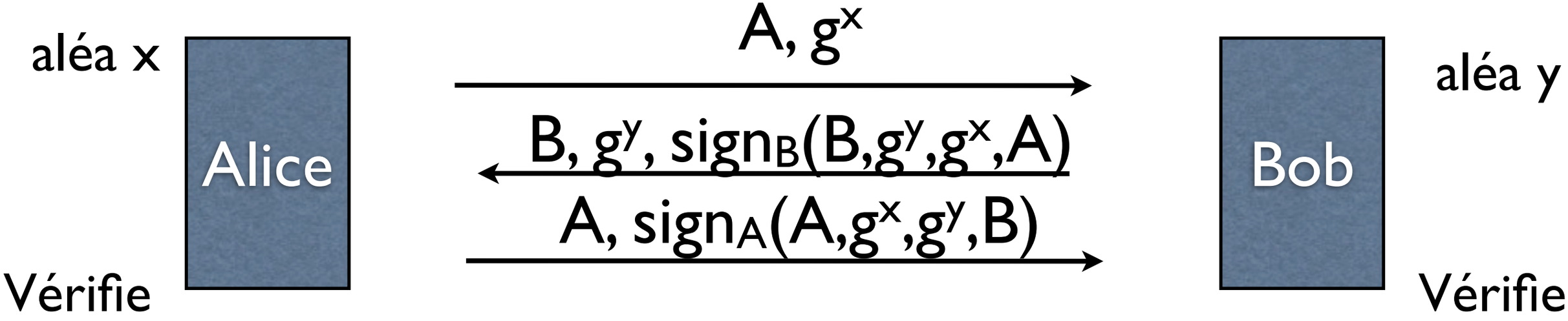


Schéma hybride

Comment chiffrer efficacement de longs messages sans avoir de clé en commun ?

Avantage / inconvénient des systèmes symétriques et asymétriques

Clé asymétrique:

- Avantages:
  - Gestion des clés (seule la clé secrète doit le rester), (*n* clés)
  - Non-répudiation
- Inconvénients:
  - Lenteur (100 fois plus lent, dépend de la taille de du module)
  - Charge machine importante

Clé symétrique:

- Avantages:
  - Rapidité (soft qq 10Mo/s, hard qq 100Mo/s)
  - Clés très courtes
  - Peu gourmand en ressources machines
- Inconvénients:
  - Gestion des clés (n^2)
  - Échange préalable à toute comm.
  - Pas de non-répudiation

Tirer avantage de la cryptographie symétrique et asymétrique

Annuaire

ID || Public Key

$K_{pk\text{-}bob}$

ID_{bob}

(C_0, C_1)

Générer aléatoirement K_s

$C_0 = E_{RSA}(K_{pk\text{-}bob}, K_s)$

$C_1 = E_{AES}(K_s, M)$

$K_s = D_{RSA}(K_{sk\text{-}bob}, C_0)$ M

$= D_{AES}(K_s, C_1)$

Gestion des clés

Distribution des clés et Certificats

- Un algorithme à clé publique a besoin d'un mécanisme de distribution des clés
- Garantie que la clé utilisée est bien celle du correspondant
  ⇒ utilisation de PKI
- Infrastructure de Gestion de Clés : Public-Key Infrastructure
- Si ce mécanisme est absent, il est possible de réaliser une attaque par le milieu de tout système de chiffrement asymétrique

Attaque par le milieu

Annuaire

~~$ID_{bob} \| K_{pk\text{-}bob}$~~

$ID_{bob} \| K_{pk\text{-}oscar}$

$K_{pk\text{-}oscar}$

ID_{bob}

(C_0, C_1) (C'_0, C_1)

Générer aléatoirement K_s

$C_0 = E_{RSA}(K_{pk\text{-}oscar}, K_s)$

$C_1 = E_{AES}(K_s, M)$

$K_s = D_{RSA}(K_{sk\text{-}bob}, C_0)$

$M = D_{AES}(K_s, C_1)$

$K_s = D_{RSA}(K_{sk\text{-}oscar}, C_0)$

$M = D_{AES}(K_s, C_1)$

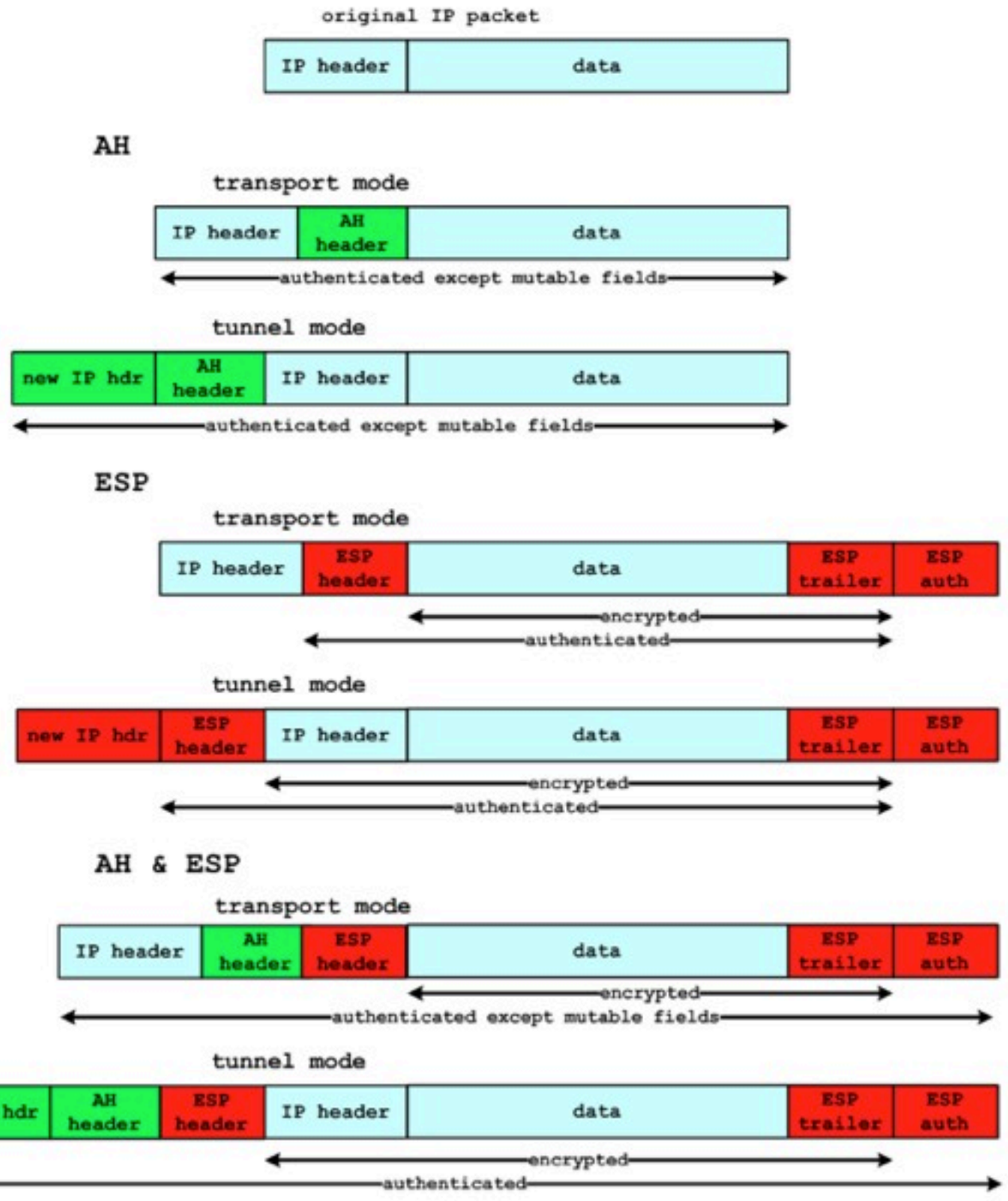
$C'_0 = E_{RSA}(K_{pk\text{-}bob}, Ks)$

Protocoles de sécurité

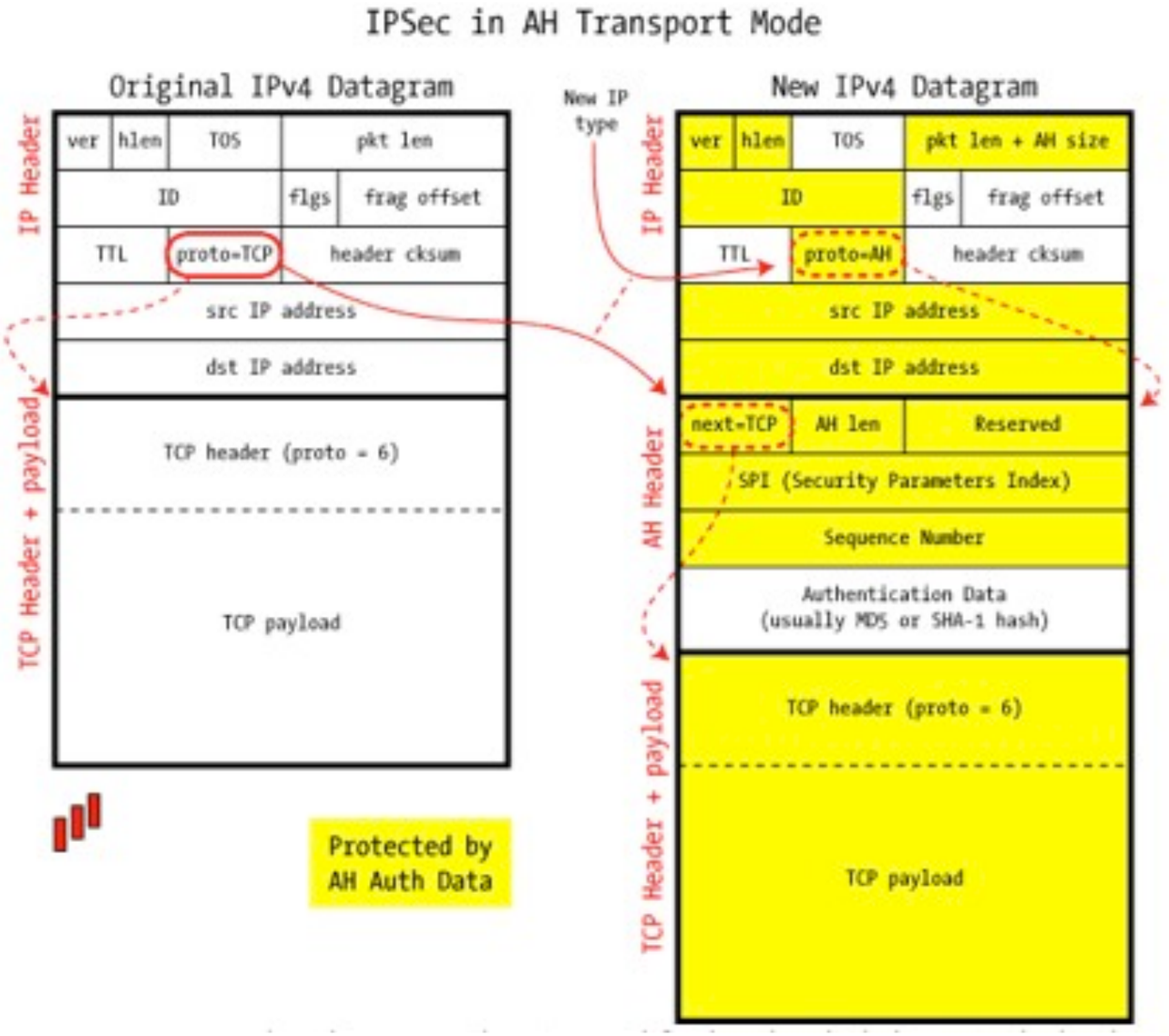
IPSEC

- Securité au niveau des paquets IP
- modes tunnel (routeur à routeur) et transport (machine à machine)
- pas de sécurité dans IPv4 alors que IPv6 prévoit de la place pour les IVs dans les headers
- AH: Authentification du header
- ESP: Authentification et chiffrement du payload et du header dans certains cas
- ISAKMP/IKE: Protocole d'échange de clé

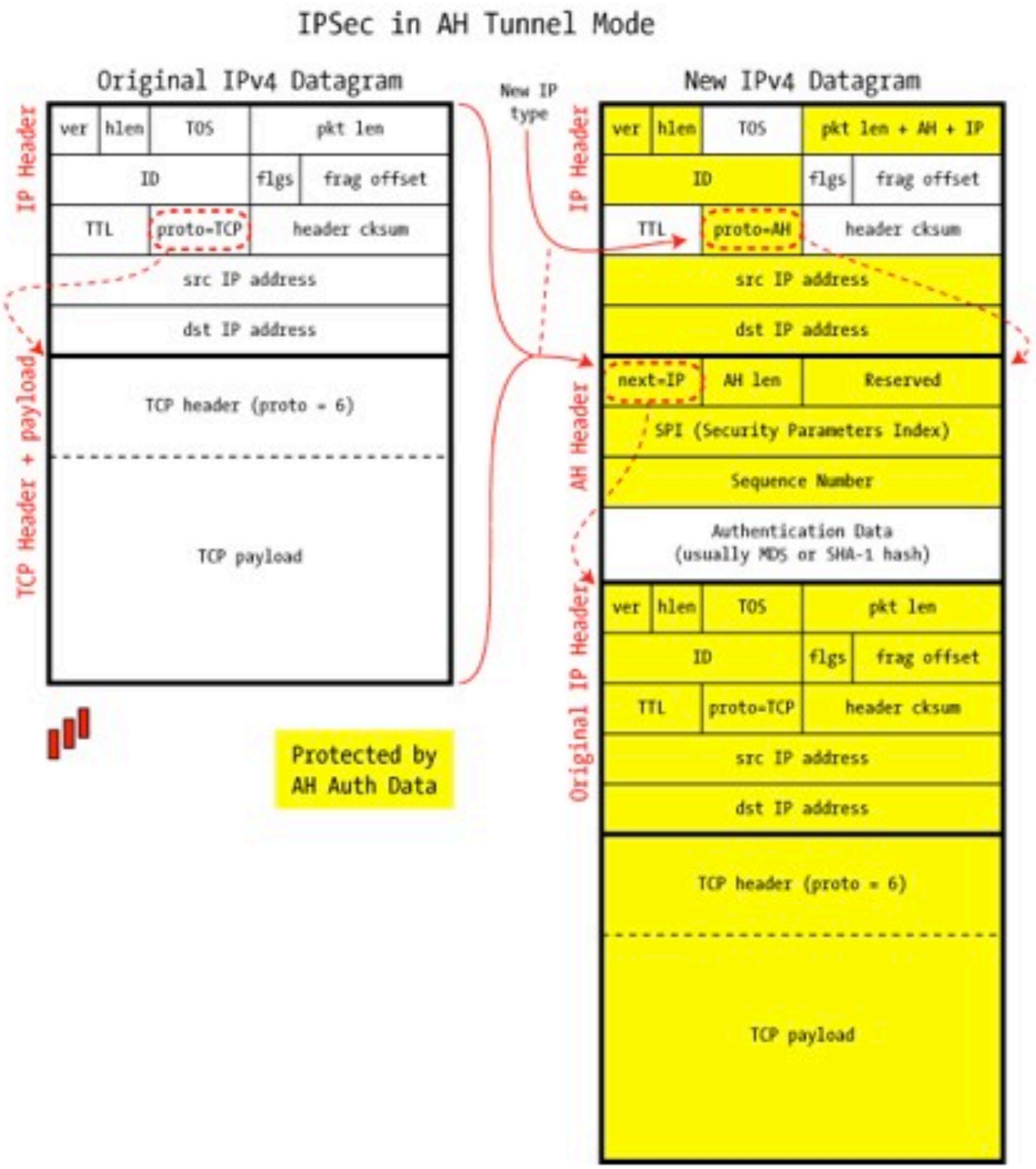
Paquets


original IP packet
IP header
data
AH
transport mode
IP header
AH header
data
authenticated except mutable fields
tunnel mode
new IP hdr
AH header
IP header
data
authenticated except mutable fields
ESP
transport mode
IP header
ESP header
data
ESP trailer
ESP auth
encrypted
authenticated
tunnel mode
new IP hdr
ESP header
IP header
data
ESP trailer
ESP auth
encrypted
authenticated
AH & ESP
transport mode
IP header
AH header
ESP header
data
ESP trailer
ESP auth
encrypted
authenticated except mutable fields
tunnel mode
new IP hdr
AH header
ESP header
IP header
data
ESP trailer
ESP auth
encrypted
authenticated

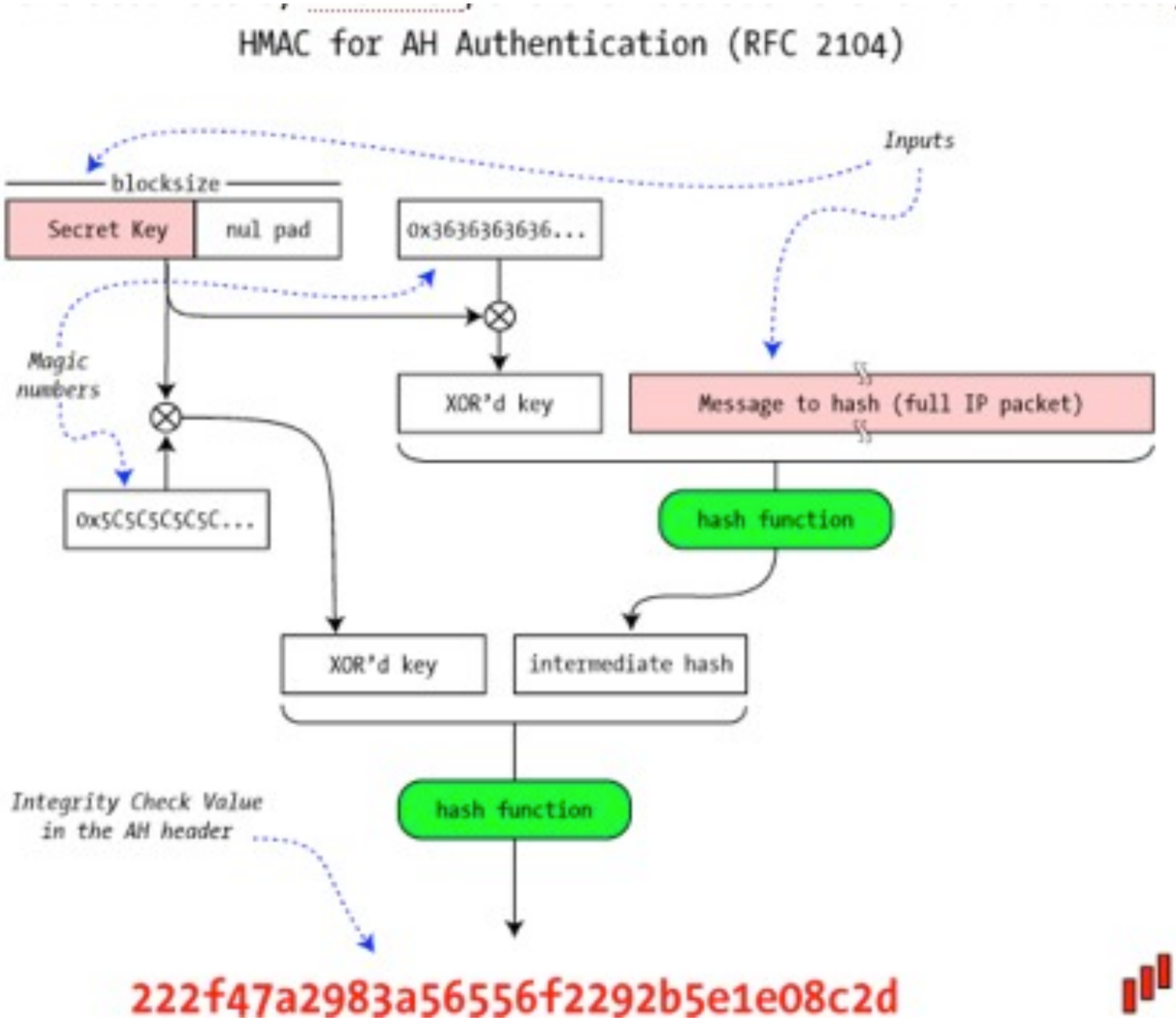
AH Transport



AH Tunnel


IPSec in AH Tunnel Mode
Original IPv4 Datagram
IP Header
TCP Header + payload
ver
hlen
TOS
pkt len
ID
flgs
frag offset
TTL
proto=TCP
header cksum
src IP address
dst IP address
TCP header (proto = 6)
TCP payload
Protected by
AH Auth Data
New IP
type
New IPv4 Datagram
IP Header
AH Header
Original IP Header
ver
hlen
TOS
pkt len + AH + IP
ID
flgs
frag offset
TTL
proto=AH
header cksum
src IP address
dst IP address
next=IP
AH len
Reserved
SPI (Security Parameters Index)
Sequence Number
Authentication Data
(usually MD5 or SHA-1 hash)
ver
hlen
TOS
pkt len
ID
flgs
frag offset
TTL
proto=TCP
header cksum
src IP address
dst IP address
TCP header (proto = 6)
TCP payload

HMAC


HMAC for AH Authentication (RFC 2104)
Inputs
blocksize
Secret Key
nul pad
0x3636363636...
Magic numbers
XOR'd key
Message to hash (full IP packet)
0x5C5C5C5C5C...
hash function
XOR'd key
intermediate hash
hash function
Integrity Check Value in the AH header
222f47a2983a56556f2292b5e1e08c2d

AH and NAT

Though AH provides very strong protection of a packet's contents because it covers *everything* that can be possibly considered immutable, this protection comes at a cost: AH is incompatible with NAT (Network Address Translation).

NAT is used to map a range of private addresses (say, 192.168.1.X) to and from a (usually) smaller set of public address, thereby reducing the demand for routable, public IP space. In this process, the IP header is actually modified on the fly by the NAT device to change the source and/or destination IP address.

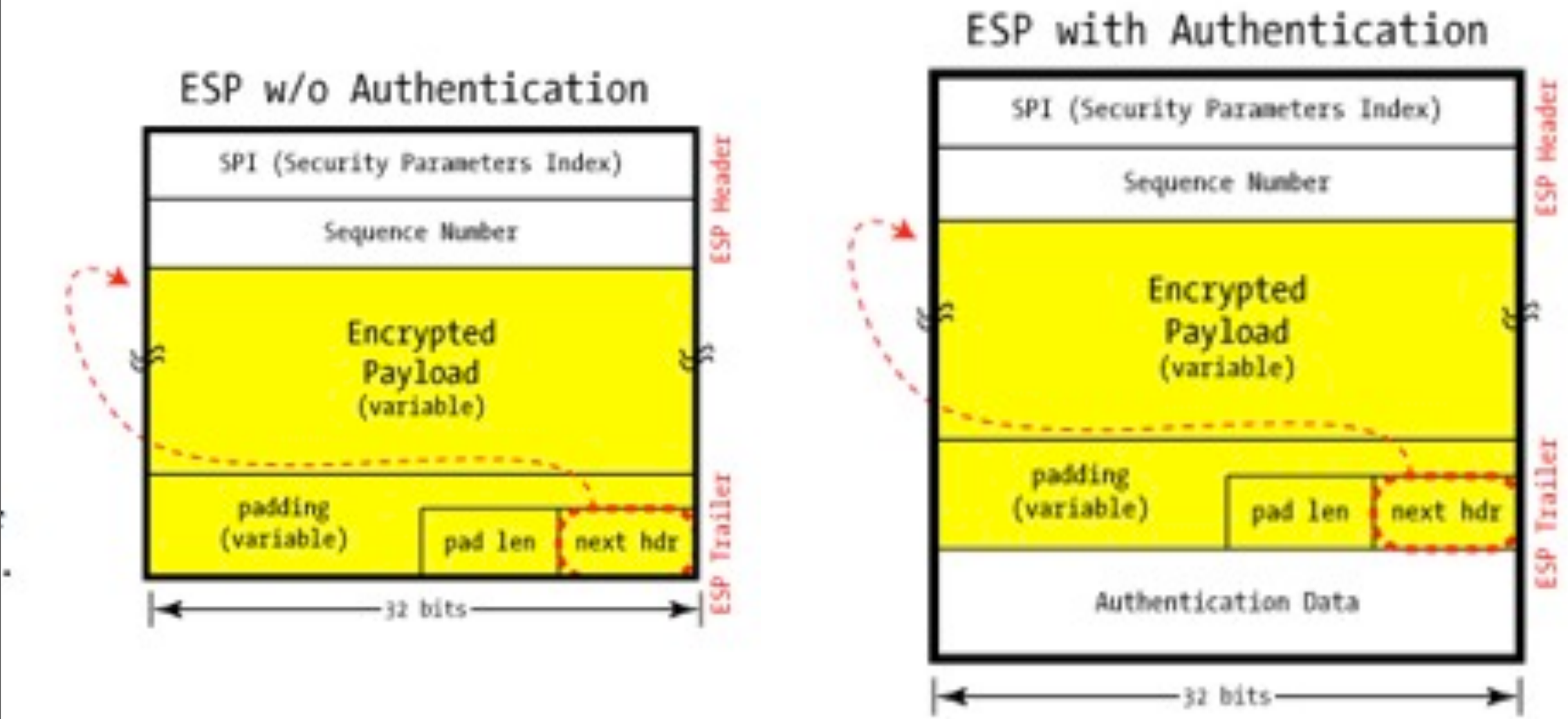
When the appropriate source or header IP address is changed, it forces a recalculation of the header checksum. This has to be done anyway, because the NAT device typically serves as one "hop" in the path from source to destination, and this requires the decrement of the TTL (Time To Live) field.

Because the TTL and header checksum fields are *always* modified in flight, AH knows to excludes them from coverage, but this does not apply to the IP addresses. These are *included* in the Integrity Check Value, and any modification will cause the check to fail when verified by the recipient. Because the ICV incorporates a secret key which is unknown by intermediate parties, the NAT router is not able to recompute the ICV.

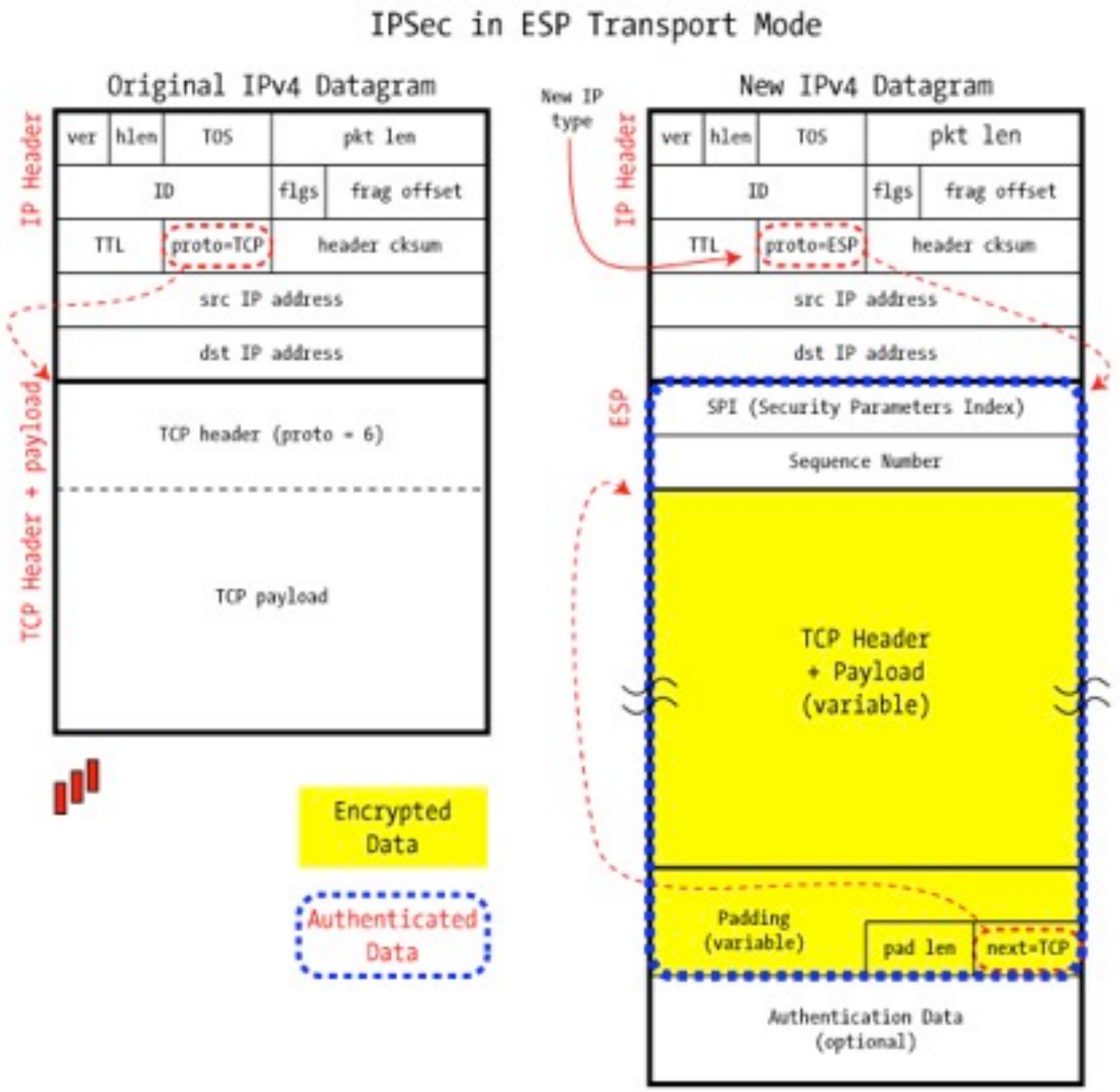
This same difficulty also applies to PAT (Port Address Translation), which maps multiple private IP addresses into a single external IP address. Not only are the IP addresses modified on the fly, but the UDP and TCP port numbers (and sometimes even to payload). This requires much more intelligence on the part of the NAT device, and more extensive modifications to the whole IP datagram.

For this reason, AH — whether in Tunnel or Transport mode — is entirely incompatible with NAT, and it may only be employed when the source and destination networks are reachable without translation.

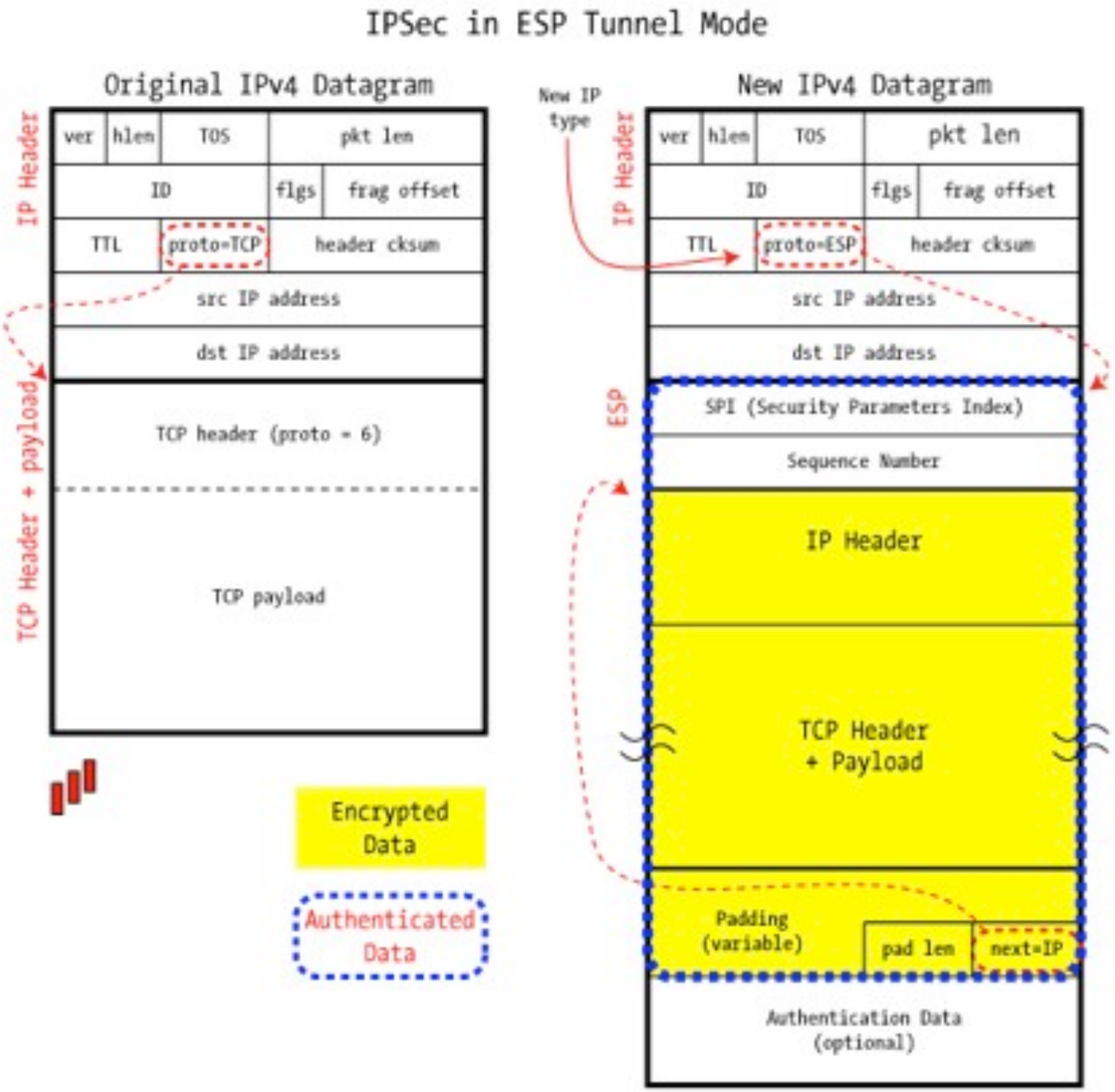
ESP: Encapsulated Security Payload



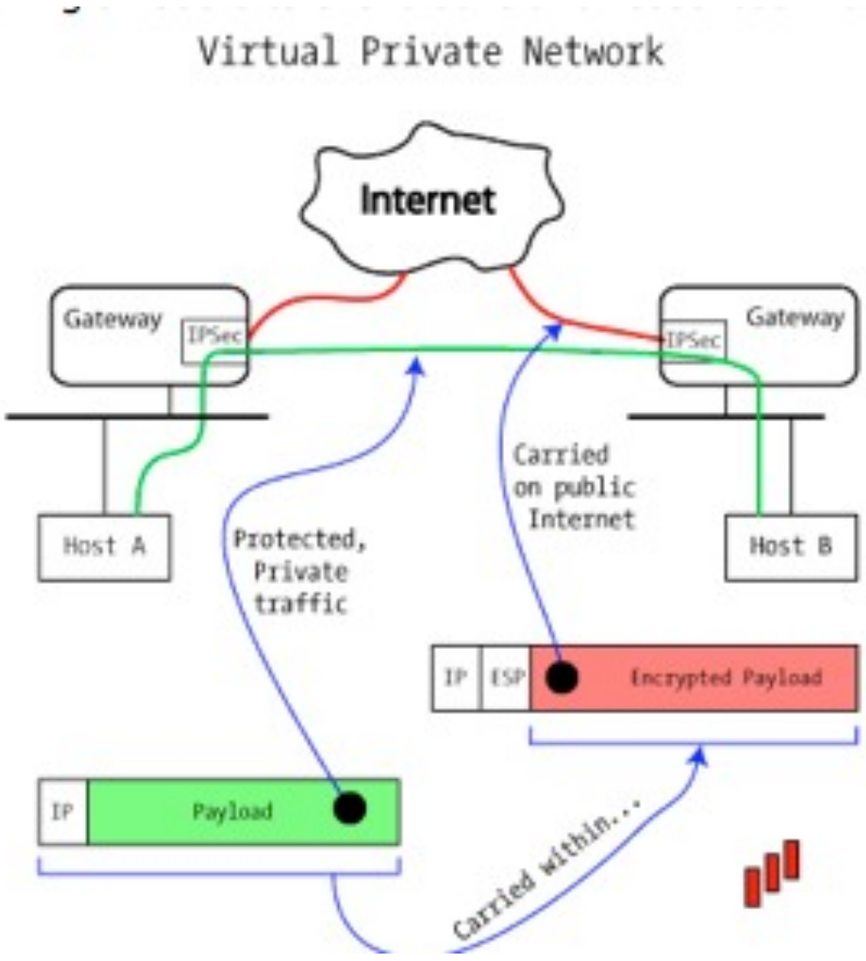
ESP Transport mode

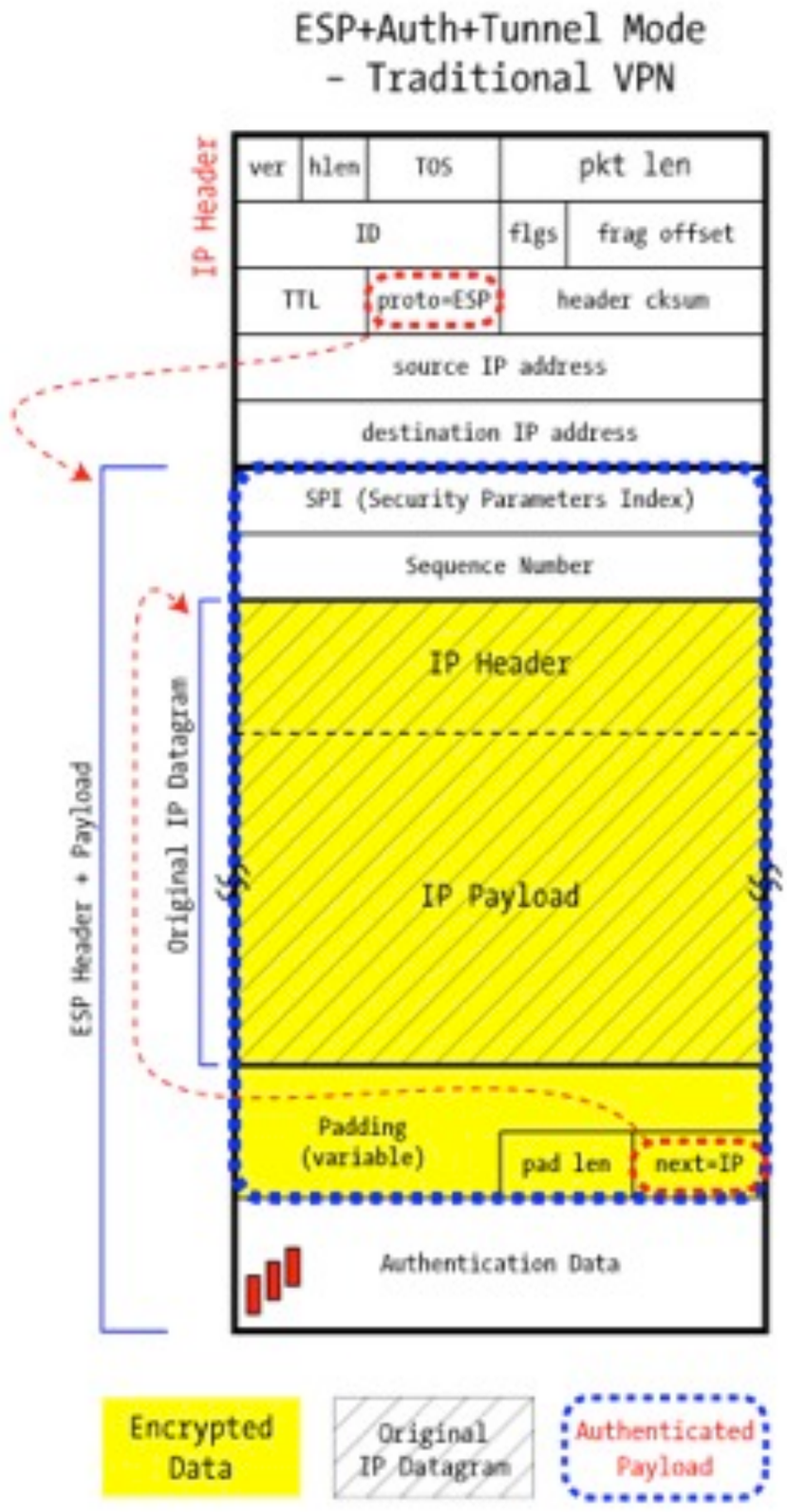


ESP Tunnel mode



ESP+AH = VPN


Virtual Private Network
Internet
Gateway
IPSec
IPSec
Gateway
Host A
Host B
Protected,
Private
traffic
Carried
on public
Internet
IP
ESP
Encrypted Payload
IP
Payload
Carried within...


ESP+Auth+Tunnel Mode
- Traditional VPN
IP Header
ver
hlen
TOS
pkt len
ID
flgs
frag offset
TTL
proto=ESP
header cksum
source IP address
destination IP address
SPI (Security Parameters Index)
Sequence Number
IP Header
IP Payload
Padding
(variable)
pad len
next=IP
Authentication Data
ESP Header + Payload
Original IP Datagram
Encrypted
Data
Original
IP Datagram
Authenticated
Payload