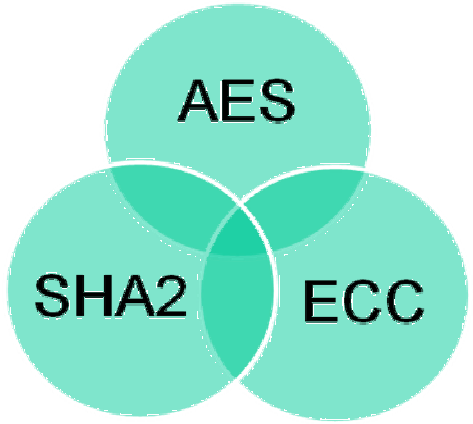
CONCEPTION D'UN CRYPTO PROCESSEUR

- ***BUT:*** *Concevoir un cryptoprocesseur implanté sur FPGA afin de crypter d'authentifier et d'assurer l'intégrité des données*

- La Confidentialité: ***Le message ne peut être lu que par le destinataire légitime*** ***(cryptage)***
- L'Authentification: ***S'assurer de l'identité de l'expéditeur*** ***(signature).***
-L'Intégrité: ***Le message n'a pas été modifié ou corrompu durant la communication*** ***(hachage)***
. Protocole de cryptage vidéo : Protocole de cryptage texte : Protocole de cryptage image :

- *Pour avoir un cryptoprocesseur générique: Traitement des données multimédias, tel que texte/image/audio/vidéo, plusieurs normes de cryptage sont utilisées.*
  - *SHA-2*
  - *AES*
  - *ECC*
  - *W7/A5/1*

- *Surface*
- *Consommation*



- *Rapidité*
- *QoSS (quality of service)*

architecture optimal

L'AES: ADVANCED ENCRYPTION STANDARD

C'est le nouveau standard de cryptage depuis 2001, il est orienté octet.

Le texte clair est de taille 128 bits (16 octets). Le texte chiffré de taille 128 bits

Les clés sont de tailles 128 bits (standard), 192 ou 256 bits (top secret)

Le texte clair est rempli dans une matrice 4x4

P0	P4	P8	P12
P1	P5	P9	P13
P2	P6	P10	P14
P3	P7	P11	P15

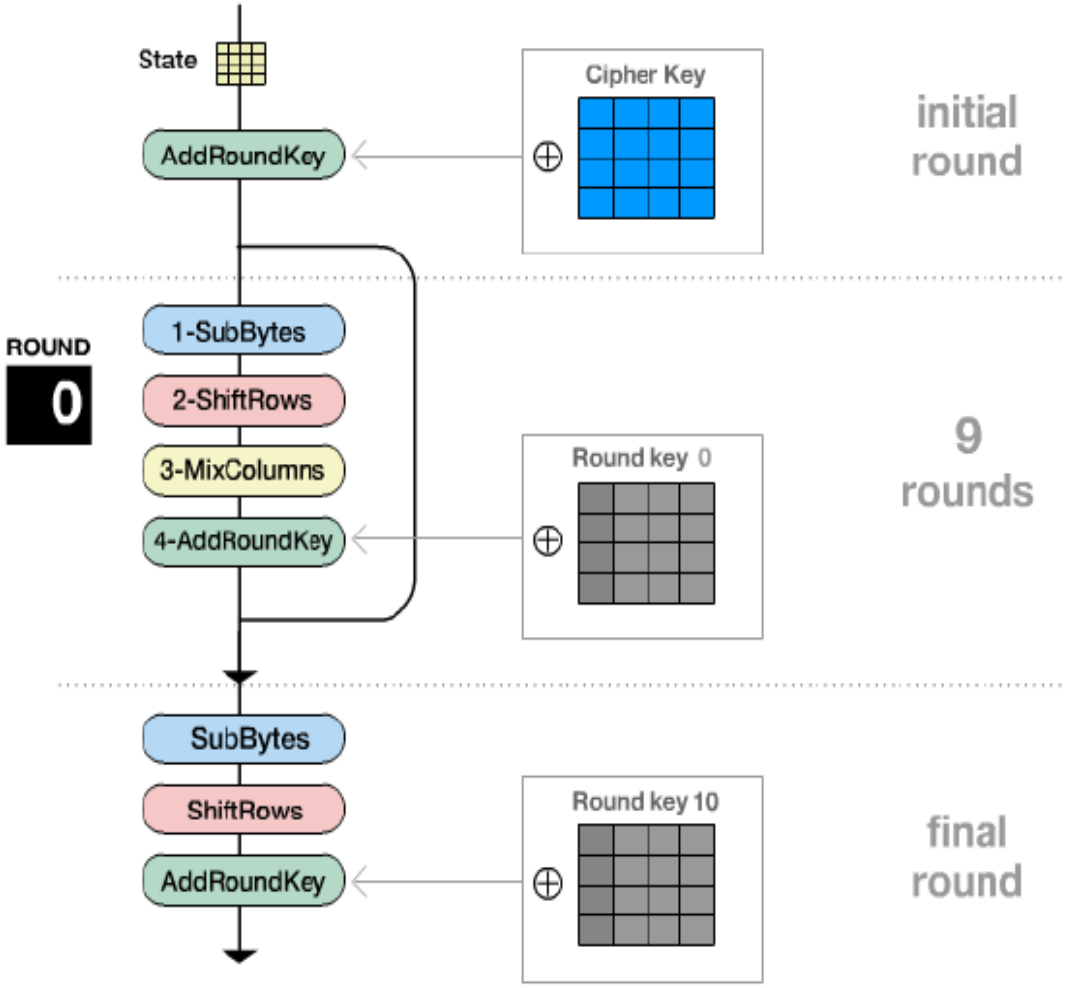
Nk: le nombre de colonne pour les clés

Nk=4 (clé 128 bits), Nk=6 (clé 192 bits), Nk=8 (clé 256 bits)

k_0	k_4	k_8	k_{12}	k_{16}	k_{20}	k_{24}	k_{28}
k_1	k_5	k_9	k_{13}	k_{17}	k_{21}	k_{25}	k_{29}
k_2	k_6	k_{10}	k_{14}	k_{18}	k_{22}	k_{26}	K_{30}
k_3	k_7	k_{11}	k_{15}	k_{19}	k_{23}	k_{27}	k_{31}

Nr le nombre de tours

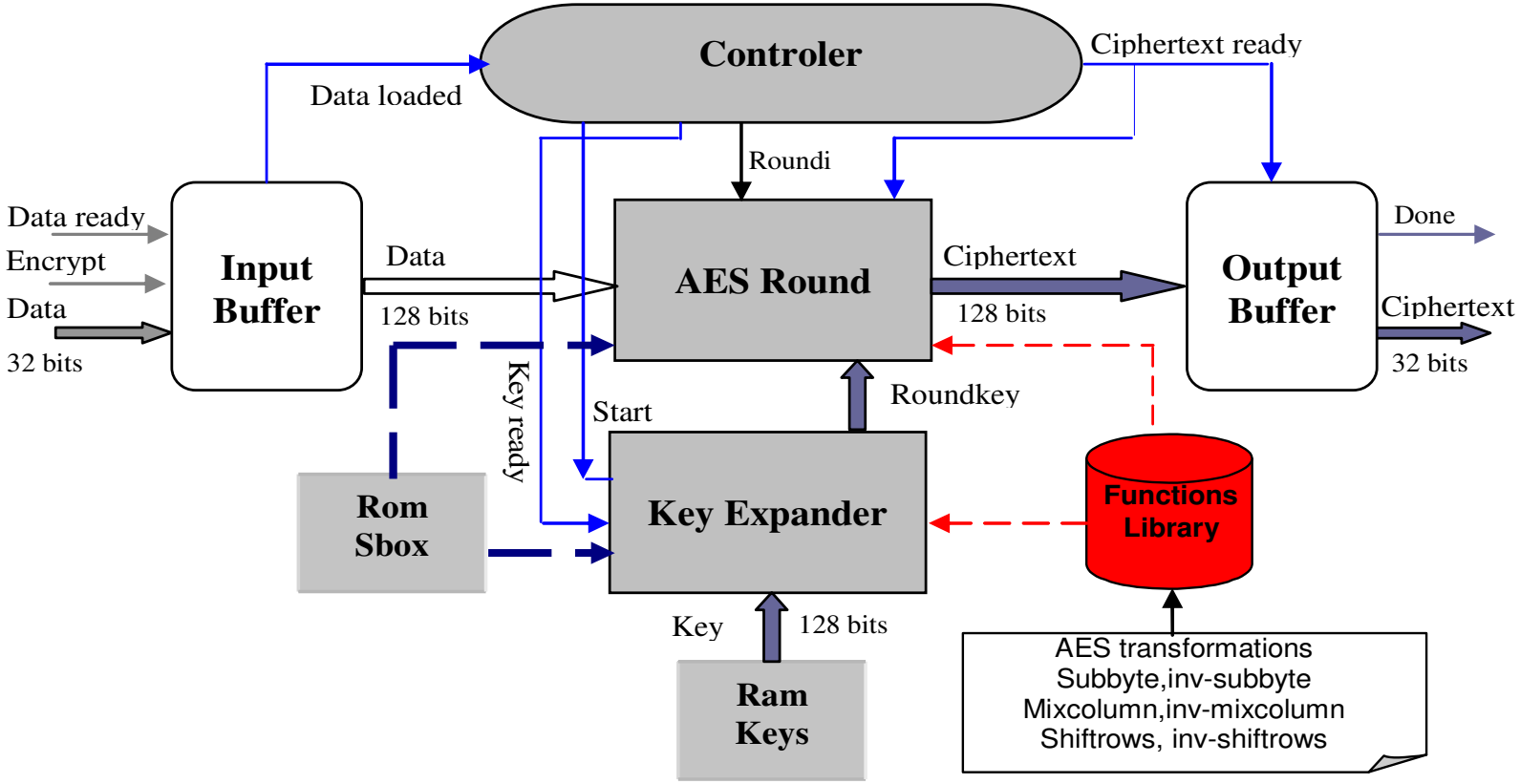
Nk	4	6	8
Nr	10	12	14



Conception d'un IP-AES

- **Séquentielle** ⟶ **Préparation des clés et stockage dans une Mémoire**
- **Parallèle** ⟶ **Préparation des clés au cours de l'opération de cryptage**

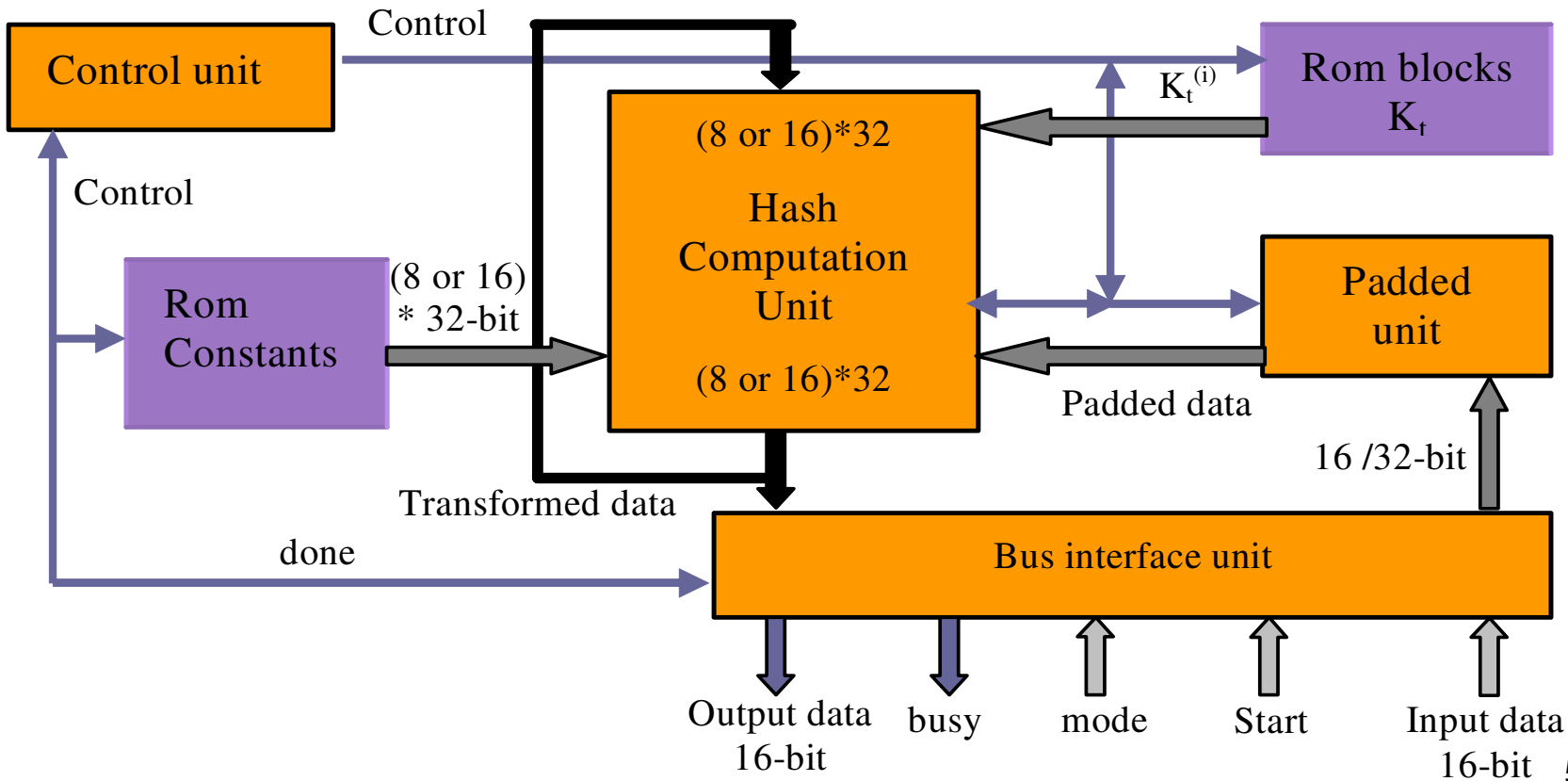
- **AES1 : Cryptage d'un bloc de 128 bits en utilisant une clé de longueur 128 bits**
- **AES2 : Cryptage/Décryptage d'un bloc de 128 bits en utilisant une clé de 128 bits**
- **AES3 : Trois variétés de l'AES (128/128, 128/192, 128/256) sur une même puce**



	Occupation (slices)	Fréquence (Mhz)	Vitesse (Gbits/s)	Consommation (mW)
AES-128	3555 out of 10752	82	1049	37,50
AES-192	3850 out of 10752	76	810	39
AES-256	4391 out of 10752	74	676	43
AES-processeur	4470 out of 10752	73	649	45
AES-processeur Modes (ECB,CBC,OFB)	5570 out of 10752	69	883	
AES-128 2-Rounds	7200 out of 10752	71	1817	
AES-128-2Rounds IP-RaM-Xilinx	5612 out of 10752	70,6	1800	

Conception d'un IP-HACHAGE

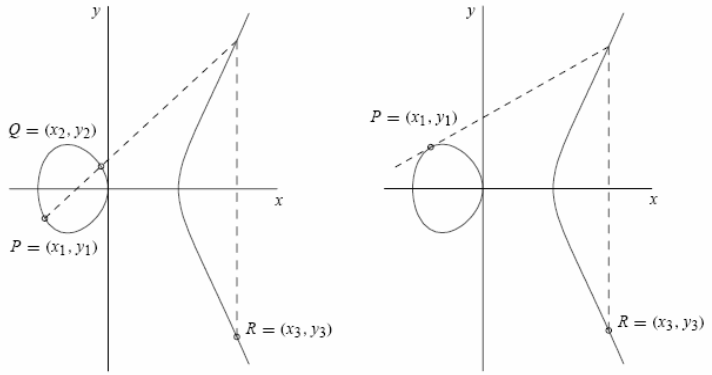
- **SHA-256** : **Entrée: message multiple de 512 bits; Sortie: haché de 256 bits.**
- **SHA-512** : **Entrée: message multiple de 1024 bits; Sortie: haché de 512 bits.**
- **SHA2-processor** : **configurable**



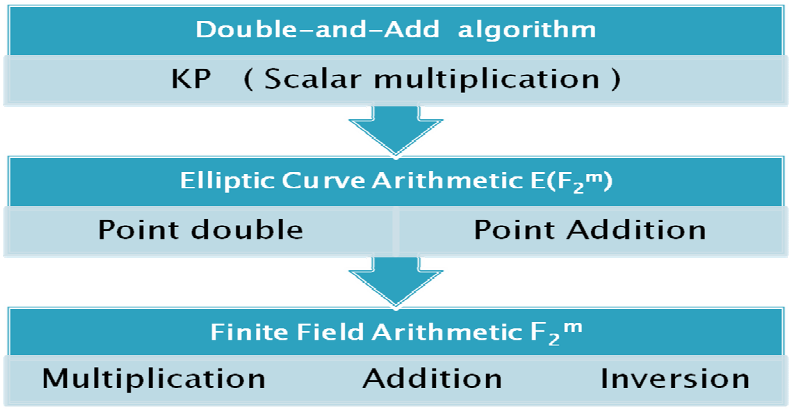
XC2v2000-6ff896	*Cycles*	*Fréquence (MHz)*	*Vitesse (Mb/s)*	*Occupation (slices)*	*Puissance (mW)*
SHA-256	**64**	**77**	**308**	**773**	**39**
SHA-512	**80**	**69**	**441,6**	**1026**	**43**
SHA-2 Processor	**64 80**	**66**	**264 : SHA-256 422 : SHA-512**	**1100**	**47**
SHA²-256	**32**	**73**	**584**	**1480**	**50**
SHA²-512	**40**	**71**	**908,8**	**2385**	**65**
SHA²-2 Processor	**32 40**	**71**	**568 : SHA-256 908: SHA-512**	**2530**	**66**

Les Courbes Elliptiques

La courbe elliptique a été proposée pour des fins cryptographiques (ECC) par Neal Koblitz et Victor Miller en 1985: $y^2 + xy = x^3 + ax^2 + b$: **avec a,b** GF_2^m



(a) Addition: $P + Q = R$. (b) Doubling: $P + P = R$.



Addition des points : $R(x_3,y_3)=P_1(x_1,y_1)+ P_2(x_2,y_2)$

$$\lambda = \frac{y_2 + y_1}{x_2 + x_1}$$

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$$

$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1$$

Doublement de point: $R(x3,y3)= 2P_1(x_1,y_1)$

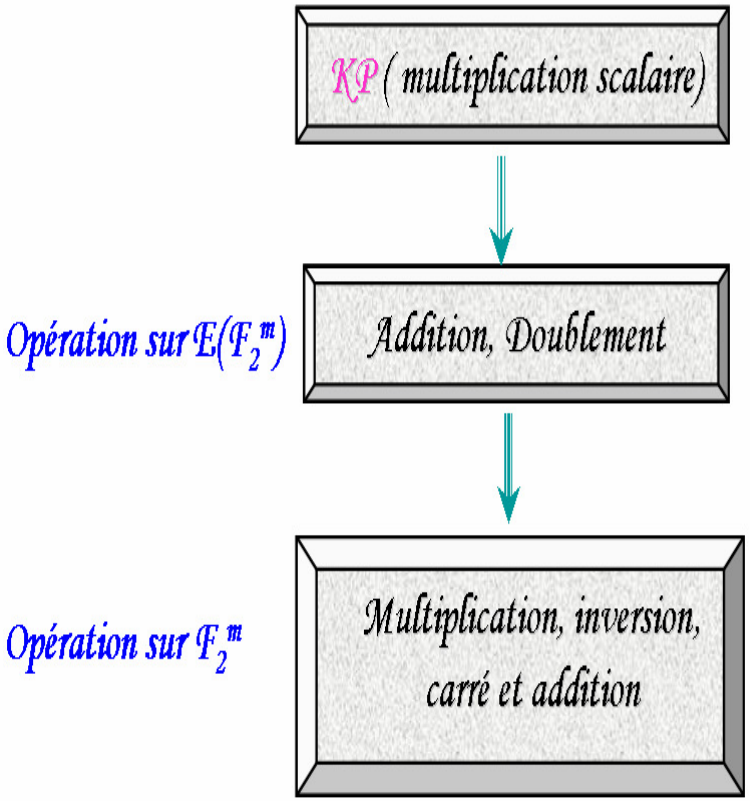
$$\lambda = x_1 + \frac{y_1}{x_1}$$

$$x_3 = \lambda^2 + \lambda + a$$

$$y_3 = (\lambda + 1)\, x_3 + x_1^2$$

La méthode de Montgomery est adoptée

Implémentation de ECC

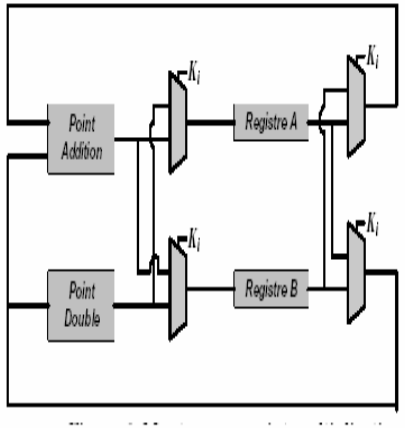


Montgomery point Multiplication

Input: $k = (k_{n-1}, k_{n-2}, \ldots, k_1, k_0)_2$ with $k_{n-1} = 1$,
$P(x, y) \in E(F2^m)$
Output: $Q = kP$

Procedure: MontPointMult (P, k)

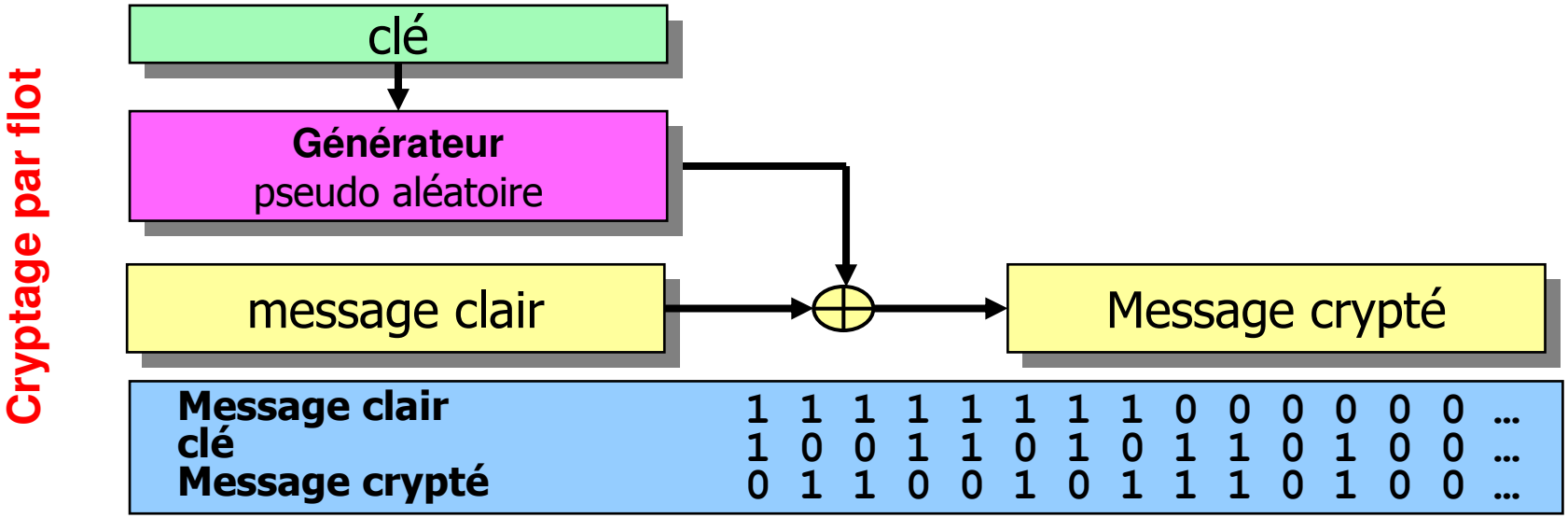
1. Set $X_1 \leftarrow x$, $Z_1 \leftarrow 1$, $X_2 \leftarrow x^4 + b$, $Z_2 \leftarrow x^2$
2. For i from n − 2 downto 0 do
   3. if ($k_i = 1$) then
      3. $(X_1, Z_1) \leftarrow$ Madd($X1, Z_1, X_2, Z_2$);
      4. $(X_2, Z_2) \leftarrow$ Mdouble(X_2, Z_2);
   4. Else
      6. $(X_1, Z_1) \leftarrow$ Madd(X_2, Z_2, X_1, Z_1);
      7. $(X_2, Z_2) \leftarrow$ Mdouble(X_1, Z_1);



5. $x3 \leftarrow X_1/Z_1$
6. $y3 \leftarrow (x + X_1/Z_1)[(X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)(Z_1Z_2)](xZ_1Z_2)^{-1} + y$
7. Return (x_3, y_3)

XC2v2000-6ff896	*Fréquence (MHz)*	*Occupation (slices)*	*Vitesse Mbps*
Inversion	**83.3**	**2373**	
Multiplication	**51.4**	1510	
ECC_point_ Multiplication	**129,90**	**12416** Out of 33792	
GPA: Grain-128	**238,5**	**310**	**238,5**

Implémentation des Générateurs pseudo-aléatoires de clés



- **La sécurité d'un système de chiffrement repose sur:**
  - ***Algorithme utilisé*: Résistance aux différentes attaques.**
  - ***Clés*: Taille des clés et qualité de leurs générations.**
- **Les méthodes de génération de clés:**
  - **Utilisation des registres à décalages (LFSRs).**
  - **Utilisations des opérations sur les courbes elliptiques (ECC).**
  - **Utilisation des automates cellulaires (CA).**
  - **Utilisation des générateurs de type :X_{n+1}= (a x_n +b) mod N**

Stream Ciphers: Caractéristiques

Avantage :

- Rapidité: Le cryptage se fait directement au niveau des registres.
- Facilité d'implémentation.
- Pas besoin de mémoire.
- Propagation des erreurs limitée ou absente.

Inconvénients:

- La qualité en termes de « randomness » du keystream généré détermine la robustesse du système.
- La réutilisation du keystream permet une cryptanalyse facile.

A5/1 stream cipher

- A5/1 est un algorithme utilisé pour assurer la confidentialité dans la norme GSM.
- A5/1 stream cipher est constitué de :

✓ Trois LFSRs (R1, R2, R3) de longueurs respectives 19, 22 et 23 qui génèrent m séquences de périodes : $2^{19} - 1$, $2^{22} - 1$, $2^{23} - 1$ respectives.

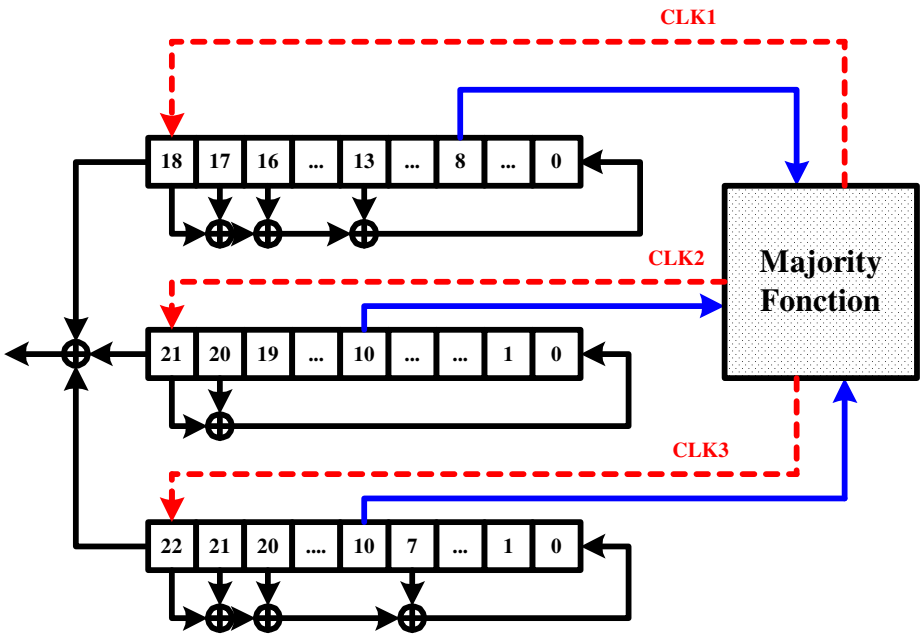
✓ Fonction majoritaire qui génère trois signaux d'horloges (Clocks) en fonction de bit 8 de R1, bit 10 de R2 et bit 10 de R3

Architecture de A5/1 stream cipher

$$LFSR\,1 : x^{19} + x^{5} + x^{2} + x + 1$$

$$LFSR\,2 : x^{22} + x + 1$$

$$LFSR\,3 : x^{23} + x^{15} + x^{2} + x + 1$$



CLK1 = CLK xor ((LFSR1(8) and (LFSR2(10) or LFSR3(10))) nor (LFSR1(8) or (LFSR2(10) and LFSR3(10))));

CLK2 = CLK xor ((LFSR2(10) and (LFSR1(8) or LFSR3(10))) nor (LFSR2(10) or (LFSR1(8) and LFSR3(10))));

CLK3 = CLK xor ((LFSR3(10) and (LFSR1(8) or LFSR2(10))) nor (LFSR3(10) or (LFSR1(8) and LFSR2(10))));

A5/2 stream cipher

- A5/2 est la version avancée de A5/1 développé en Europe en 1980.
- A5/2 est un algorithme utilisé pour assurer la confidentialité dans la norme GSM.
- A5/2 stream cipher est constitué de :

✓ Quatre LFSRs (R1, R2, R3 et R4) de longueurs respectives 19, 22, 23 et 17.

✓ Fonction majoritaire qui génère trois signaux d'horloges (Clocks) en fonction de bits 3, 7 et 10 de R4

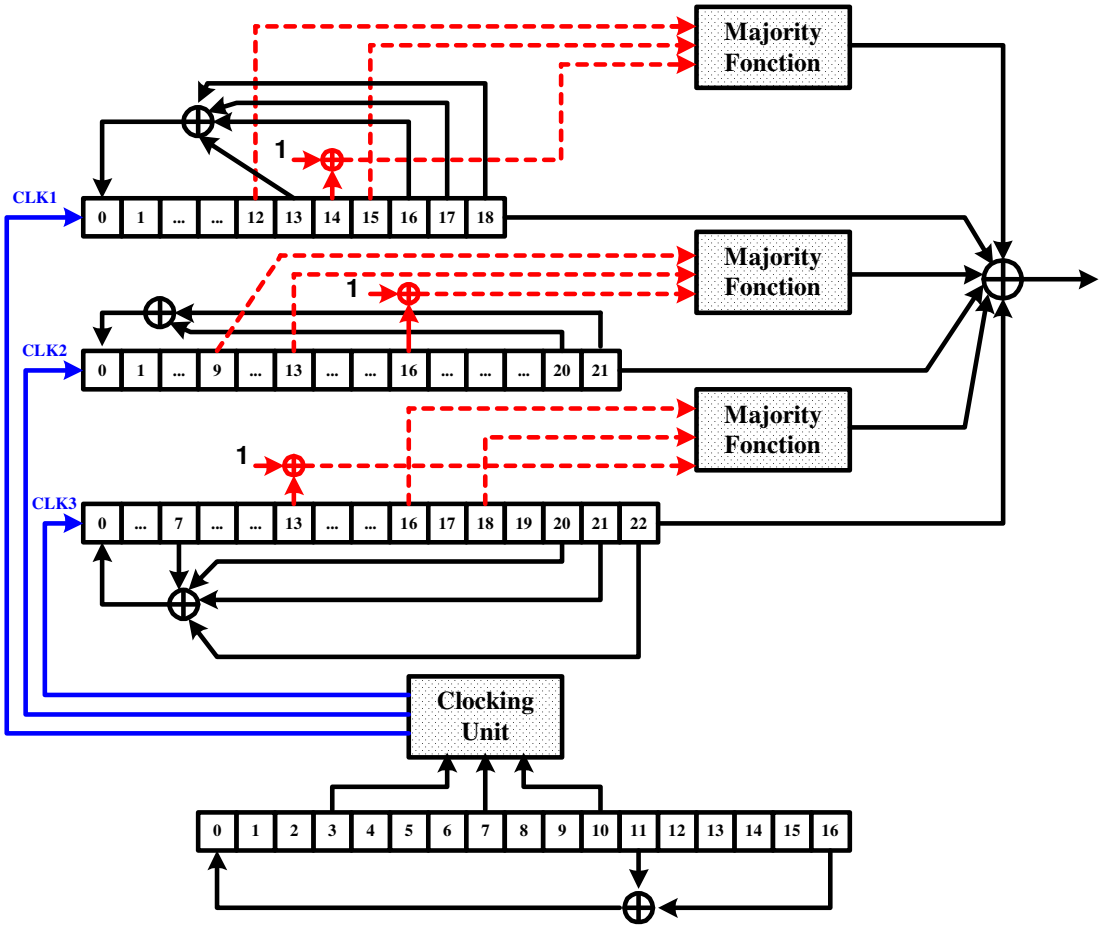
Architecture de A5/2 stream cipher

LFSR 1: $x^{19}+x^{18}+x^{17}+x^{14}+1$

LFSR 2: $x^{22}+x^{21}+1$

LFSR 3: $x^{23}+x^{22}+x^{21}+x^{8}+1$

LFSR 4: $x^{17}+x^{12}+1$



W7 stream cipher

- W7 est un algorithme de chiffrement par flot publié en Avril 2002.
- W7 stream cipher est constitué de 8 blocs identiques et synchrones
- Chaque bloc de W7 est constitué de :

✓ Trois LFSRs de longueurs respectives 38, 43 et 47 qui génèrent m-séquences de périodes : $2^{38} - 1$, $2^{43} - 1$, $2^{47} - 1$ respectives.

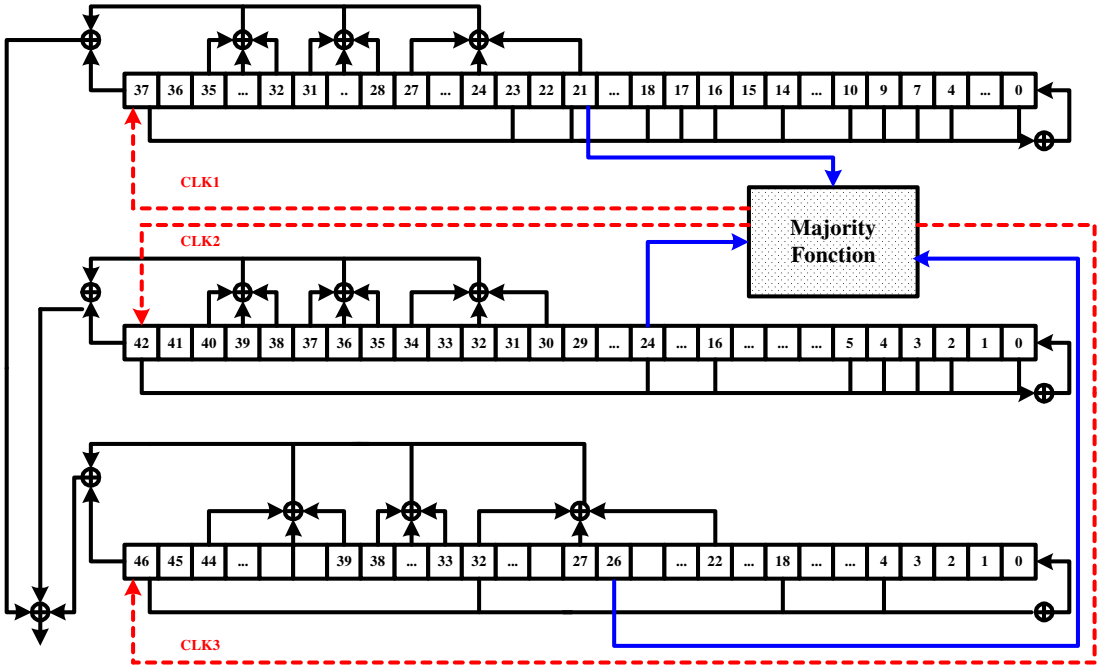
✓ Fonction majoritaire qui génère trois signaux d'horloges (Clocks) en fonction de bit 21 de R1, bit 24 de R2 et bit 26 de R3

Architecture de W7 stream cipher

LFSR1: $x^{37}+x^{24}+x^{21}+x^{20}+x^{19}+x^{18}+x^{5}+x^{2}+x+1$

LFSR2: $x^{42}+x^{21}+x+1$

LFSR3: $x^{46}+x^{39}+x^{26}+x^{25}+x^{24}+x^{23}+x^{16}+x^{3}+x^{2}+x+1$



CLK1=CLK xor ((LFSR1(21) and (LFSR2(24) or LFSR3(26))) nor (LFSR1(21) or (LFSR2(24) and LFSR3(26))));

CLK2=CLK xor ((LFSR2(24) and (LFSR1(21) or LFSR3(26))) nor (LFSR2(24) or (LFSR1(21) and LFSR3(26))));

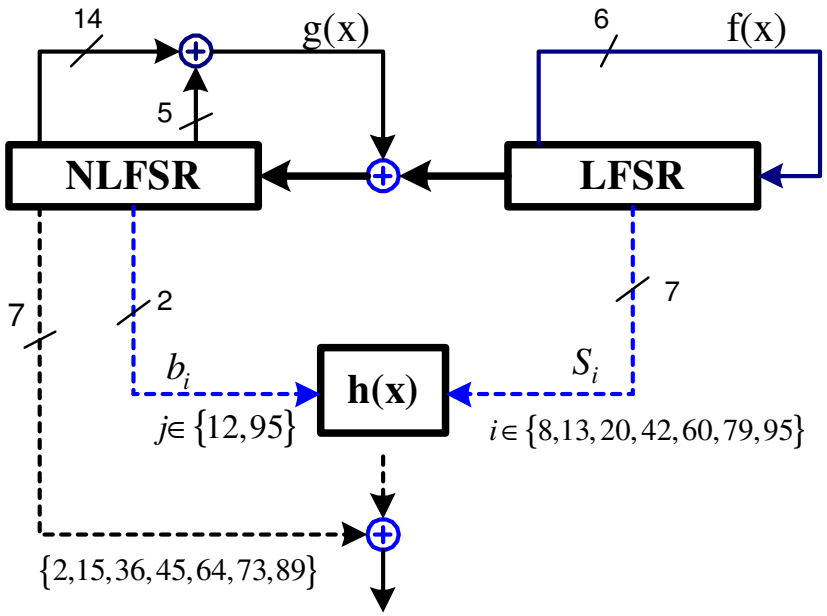
CLK3=CLK xor ((LFSR3(26) and (LFSR1(21) or LFSR2(24))) nor (LFSR3(26) or (LFSR1(21) and LFSR2(24))));

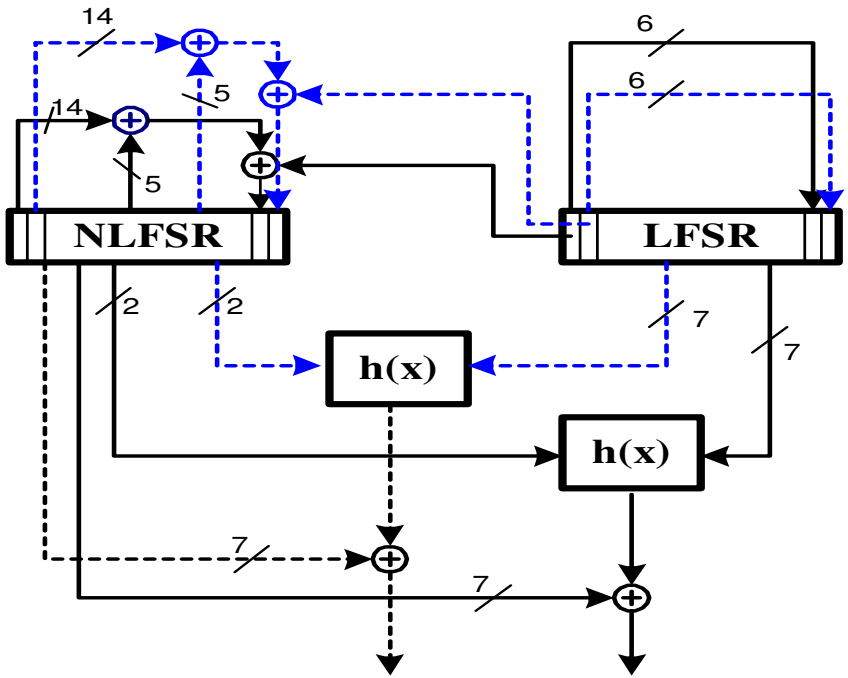
Grain stream cipher

- Grain est un algorithme de chiffrement par flot
- La première version publiée en 2006 par *Martin Hell.* Elle est constituée par
  - ✓Deux registres de 80 bits: Un LFSR + Un NLFSR
  - ✓Une fonction de sortie h(x)
  - ✓Taille de clés de 80 bits et une valeur initiale de 64 bits

- La deuxième version de Grain publiée en 2008. Elle est constituée par
  - ✓Deux registres de 128 bits: Un LFSR + Un NLFSR
  - ✓Une fonction de sortie h(x)
  - ✓Taille de clés de 128 bits et un valeur initial de 96 bits

Architecture de Grain stream cipher





$$h(x) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8$$

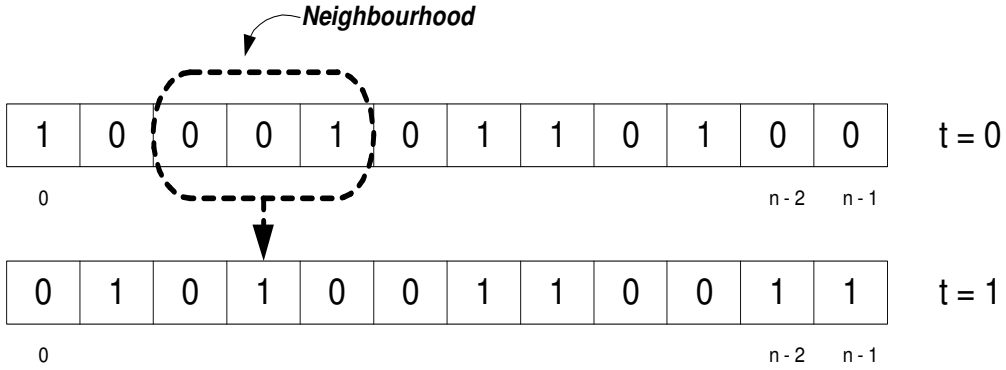
$$f(x) = 1 + x^{32} + x^{47} + x^{58} + x^{90} + x^{121} + x^{128}$$

$$g(x) = 1 + x^{32} + x^{37} + x^{72} + x^{102} + x^{128} + x^{44}x^{x60} + x^{61}x^{125} + x^{63}x^{67} + x^{69}x^{101}$$
$$+ x^{80}x^{88} + x^{110}x^{111} + x^{115}x^{117}.$$

Automates cellulaires de dimension un

- Pour un AC de dimension 1 on définit 16 rules (combinaison logiques): 0, 15, 51, 60, 85, 90, 102, 105, 150, 153, 165, 170, 195, 204, 240 et 255.

- L'état de chaque cellule i à l'instant t+1 est donner par les états des cellules i-1, i et i+1 à l'instant t.

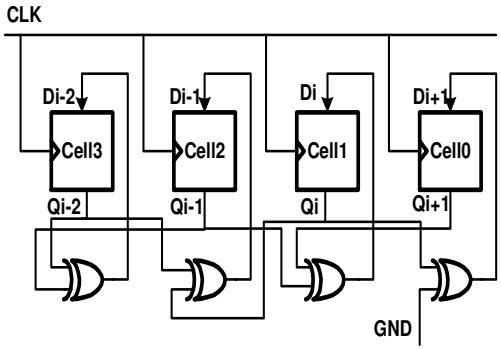


Rule 90 = 01011010

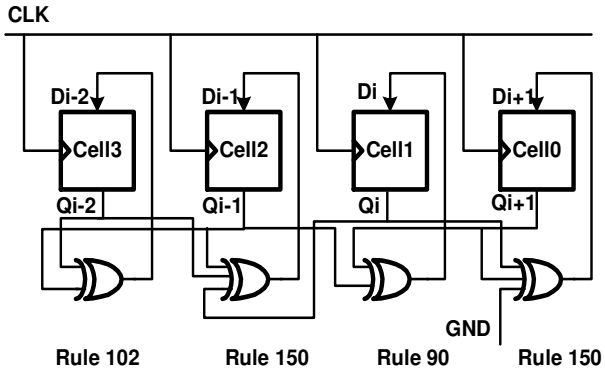
Number	7	6	5	4	3	2	1	0
Neighbourhood	111	110	101	100	011	010	001	000
Rule result	0	1	0	1	1	0	1	0

$$s_i(t+1) = s_{i-1}(t) \textbf{ xor } s_{i+1}(t).$$

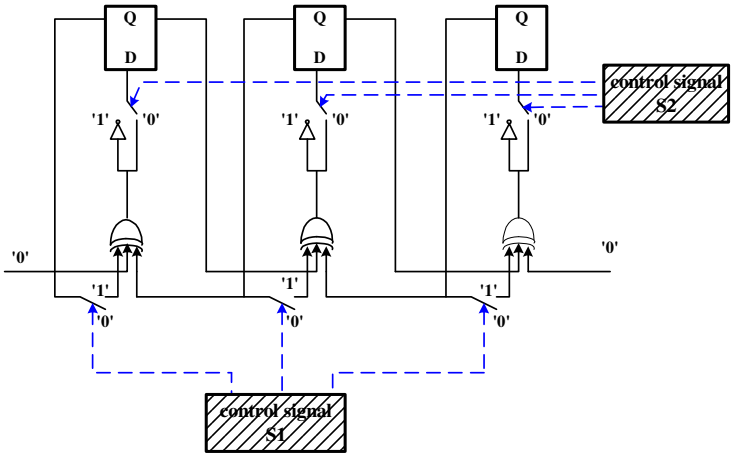
Types des AC de dimension un



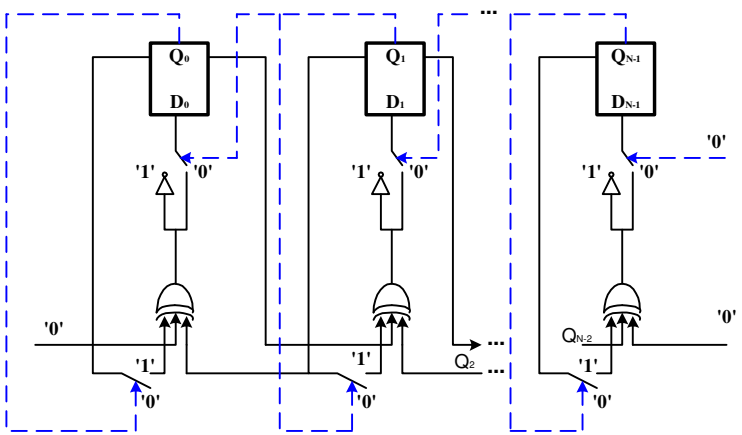
(a) AC Uniforme



(b) AC hybride

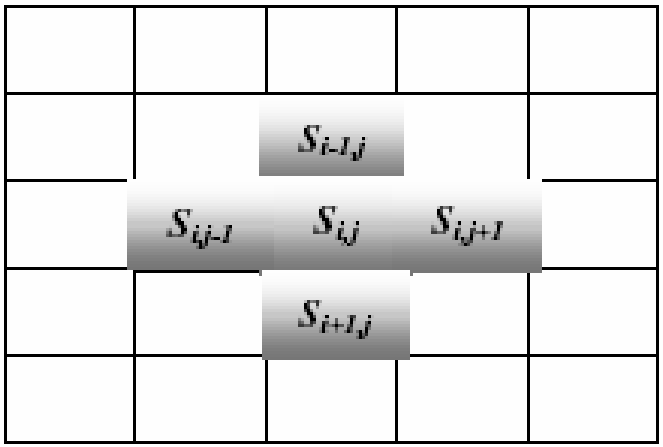


(c) AC Programmable



(d) Auto Programmable AC

Automates cellulaires de dimension deux



$$S_{i,j}(t+1) = X \text{ xor } (C \text{ and } S_{i,j}(t)) \text{ xor } (N \text{ and } S_{i-1,j}(t)) \text{ xor } (W \text{ and } S_{i,j-1}(t)) \text{ xor } (S \text{ and } S_{i+1,j}(t)) \text{ xor } (E \text{ and } S_{i,j+1}(t)).$$

X,C (center), N (North), S (South), W (west) et E (Est).

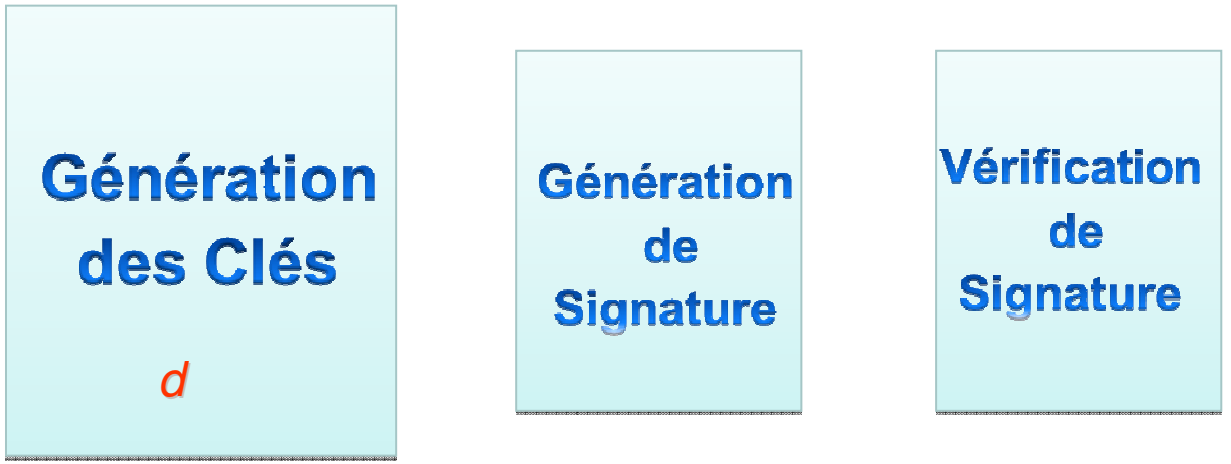
• Chaque fonction de transition (rule) est définie par (XCNWSE), exemple rule 15 (001111): $S_{i,j}(t) = S_{i-1,j}(t)$ xor $S_{i,j-1}(t)$ xor $S_{i+1,j}(t)$ xor $S_{i,j+1}(t)$

Résultats d'implémentation de stream ciphers

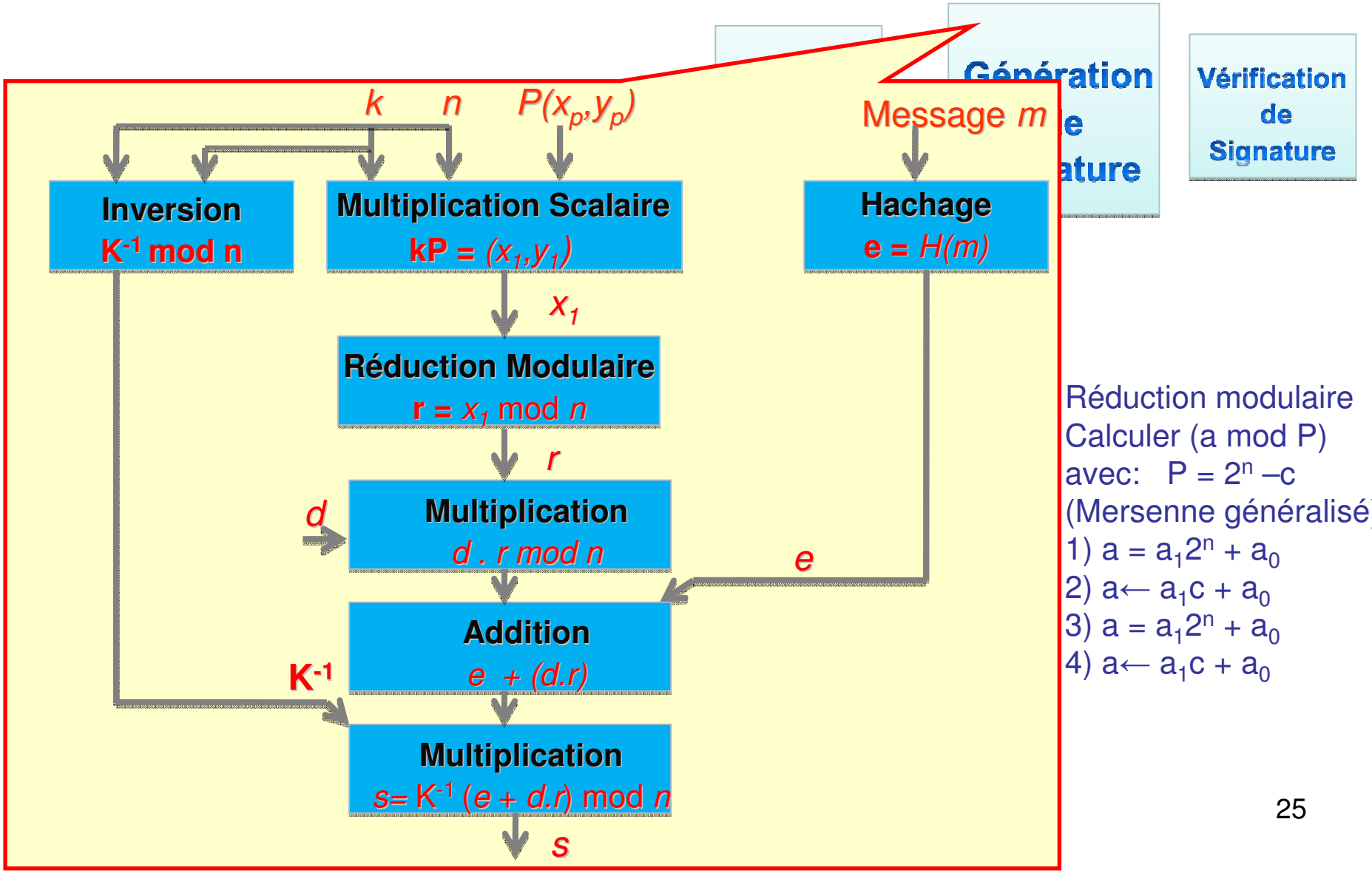
(***Virtex II***: 2V250 FG256)

Générateur		*Occupation (Slices)*	*Fréquence (Mhz)*	*Vitesse (Mbps)*	*Puissance (mw)*
A5/1		*48 (2 %)*	*391*	*391*	*33.96*
A5/2		*72 (4 %)*	*213*	*213*	*22.56*
W7		*656 (42 %)*	*193*	*1545*	*99.33*
1-D Automate Cellulaire		*198 (12 %)*	*203*	*1624*	*37.46*
Grain 128	Sortie simple	*146 (9 %)*	*325*	*325*	*61.58*
	Sortie double	*152 (9 %)*	*298*	*596*	*65.19*

Génération des Clés (ECDSA)



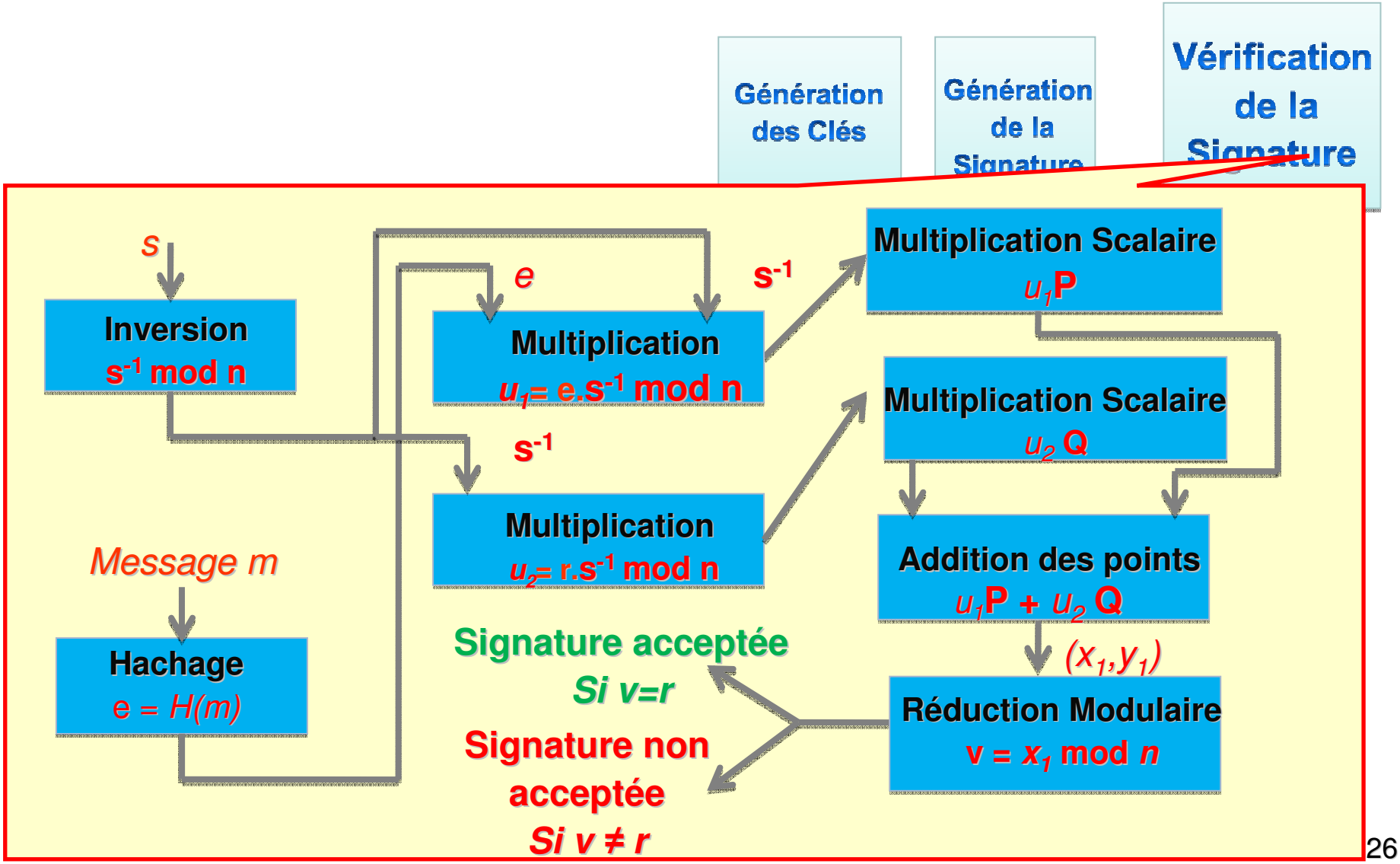
Génération de signature (ECDSA)

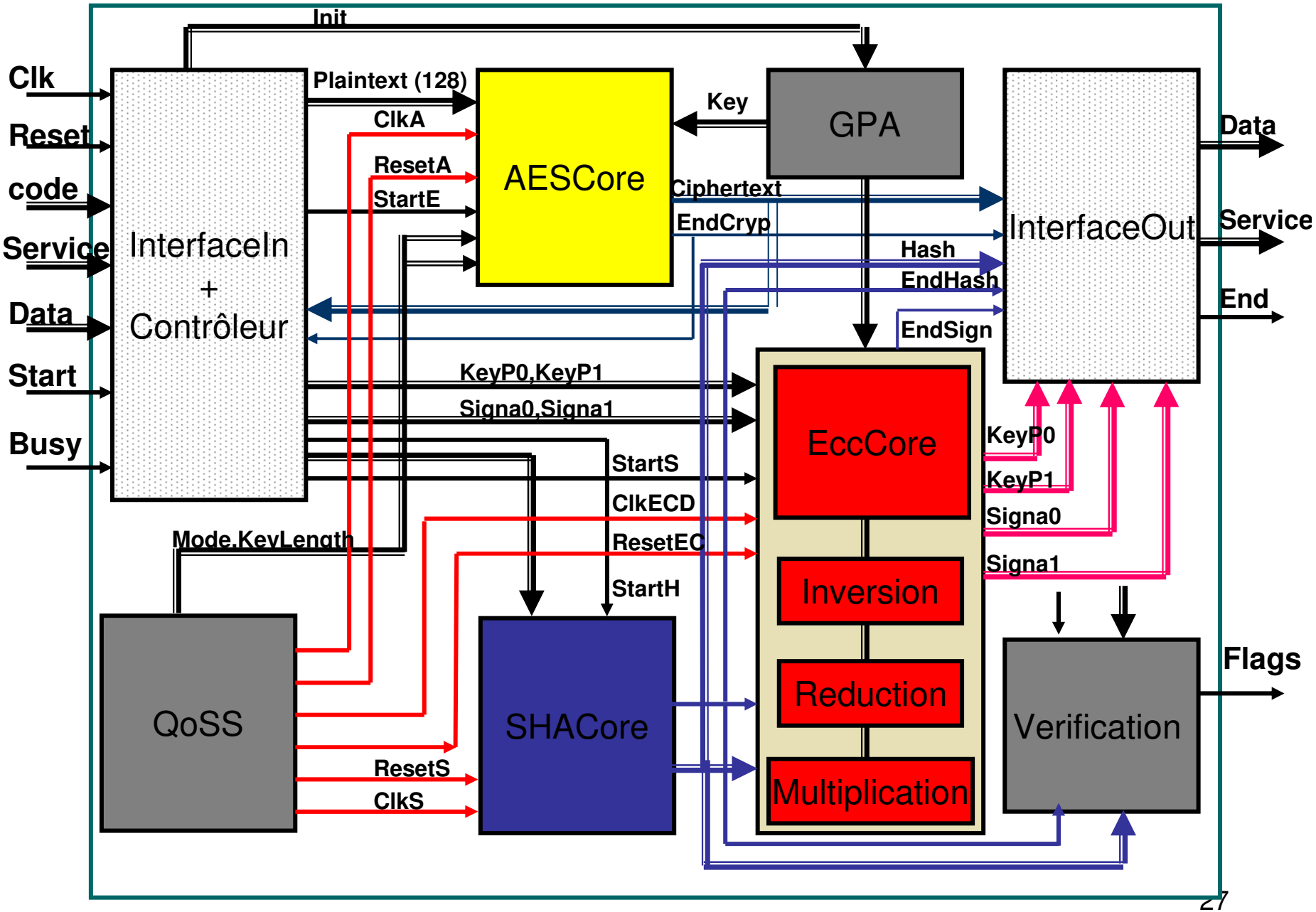


Réduction modulaire
Calculer (a mod P)
avec: $P = 2^n - c$
(Mersenne généralisé)

1) $a = a_1 2^n + a_0$
2) $a \leftarrow a_1 c + a_0$
3) $a = a_1 2^n + a_0$
4) $a \leftarrow a_1 c + a_0$

Vérification de signature (ECDSA)




Init
Clk
Reset
code
Service
Data
Start
Busy
InterfaceIn
+
Contrôleur
Plaintext (128)
ClkA
ResetA
StartE
AESCore
Key
GPA
Ciphertext
EndCryp
InterfaceOut
Data
Service
End
Hash
EndHash
EndSign
KeyP0,KeyP1
Signa0,Signa1
StartS
ClkECD
ResetEC
StartH
Mode,KeyLength
EccCore
KeyP0
KeyP1
Signa0
Signa1
Inversion
Reduction
Multiplication
QoSS
SHACore
ResetS
ClkS
Verification
Flags

Mode	Scénarios	AES	GPA	SHA-256	ECC	ECDSA
0000	Confidentailité	1	1	0	0	0
0001	Intégrité	0	0	1	0	0
0010	Authentification	0	1	1	1	1
0011	Confidentailité + Intégrité	1	1	1	0	0
0100	Confidentailité + Authentification	1	1	1	1	1
0101	Décryptage	1	1	0	0	0
0110	Vérifcation Intégrité	0	0	1	0	0
0111	Vérifcation Authentification	0	0	1	1	1
1000	Décryptage + Véerifcation Intégrité	1	1	1	0	0
1001	Décryptage + Vérifcation Authentification	1	1	1	1	1

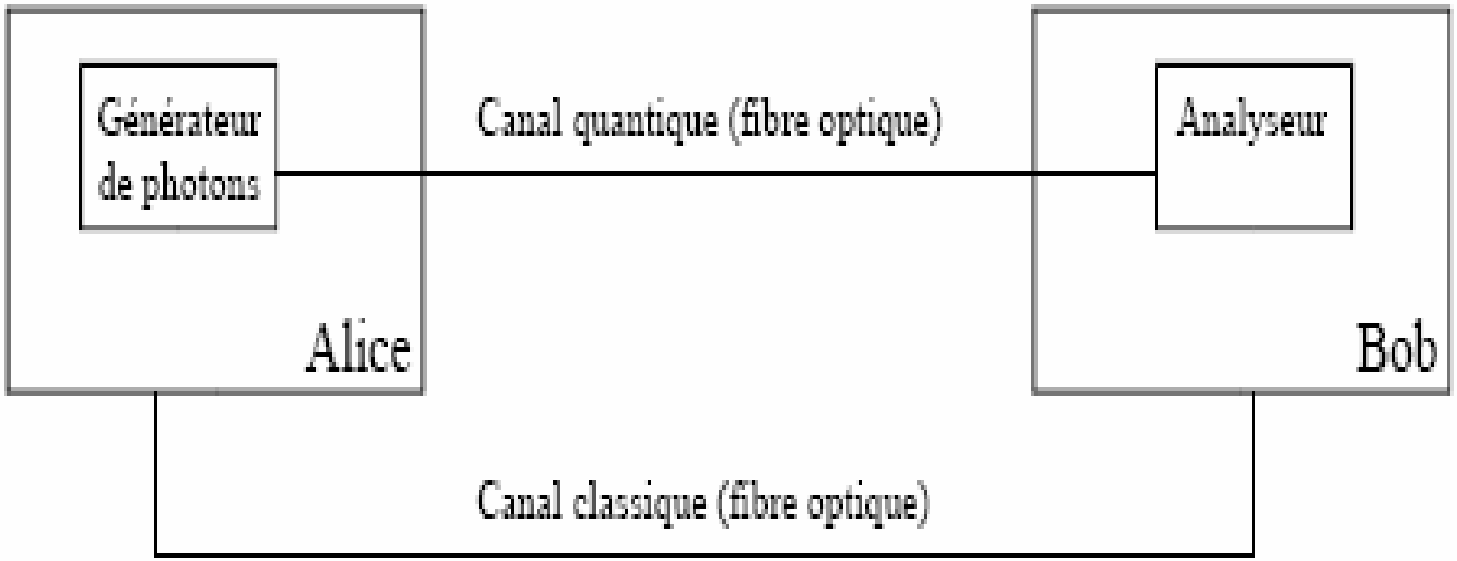
Problème d'échange de clés

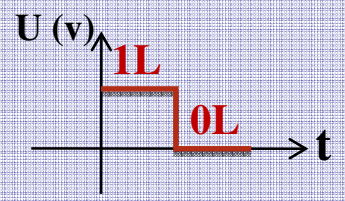
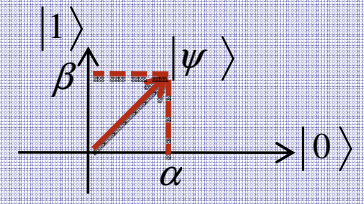
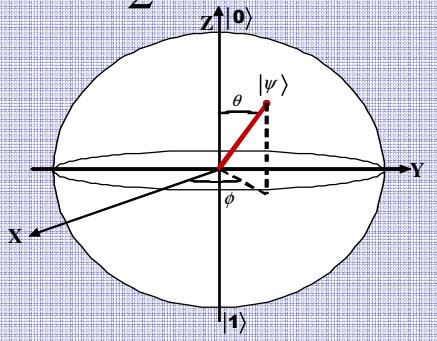
Clé publique : g, p

Alice
Clé privée : a
1 Alice calcule g^a mod p
2 Alice envoie à Bob g^a
3 Alice calcule k = $(g^b)^a$ mod p

Bob
Clé privée : b
1 Bob calcule g^b mod p
2 Bob envoie à Alice g^b
3 Bob calcule k = $(g^a)^b$ mod p

Clé commune k = g^{ab} mod p



Bit classique	Bit quantique
	1. $\|\psi\rangle = \alpha\|0\rangle + \beta\|1\rangle \Rightarrow \|\alpha\|^2 + \|\beta\|^2 = 1$  2. $\|\psi\rangle = \cos\frac{\theta}{2}\|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}\|1\rangle$ 
1L et 0L sont parfaitement définis	Une infinité d'états entre $\|0\rangle$ et $\|1\rangle$
Niveau de tension ou de courant	Des photons ou particules à spins
Algèbre de Boole	Mécanique quantique

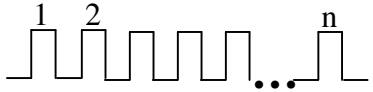
CONCEPTS DE BASE DE LA MÉCANIQUE QUANTIQUE

- Un *ket* est représenté par une matrice colonne : $|\psi\rangle = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \end{pmatrix}$
- Un *Bra* est représenté par une matrice ligne : $\langle\psi| = \begin{pmatrix} C_1^* & C_2^* & \cdots & C_n^* \end{pmatrix}$

- Un état est décrit par un vecteur ket $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$
- Calcul d'opérateur densité $\rho = |\psi\rangle\langle\psi|$

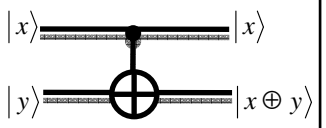
- Calcul d'entropie $S(\rho) = -\mathrm{Tr}(\rho \log_2 \rho)$
- Le registre quantique

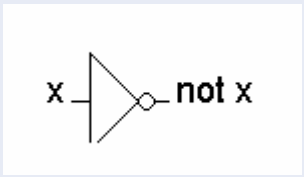
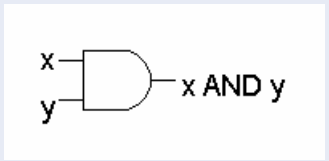
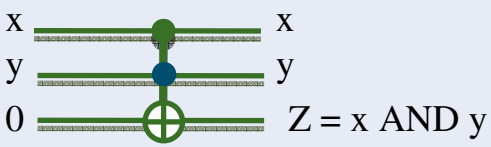
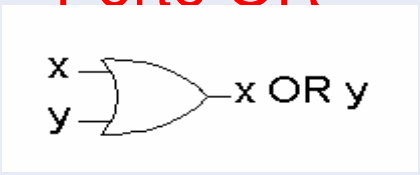
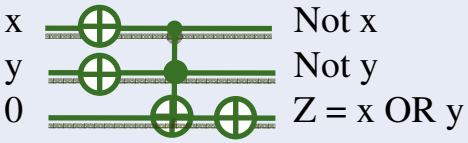
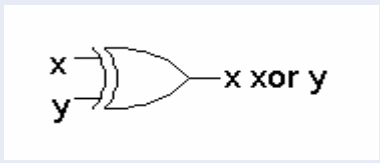
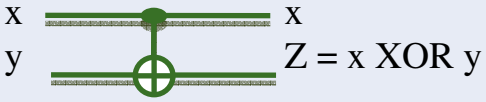
$|\psi\rangle = |\psi_1\rangle \otimes \ldots \otimes |\psi_n\rangle$: Produit tensoriel d'état



: Train de photons

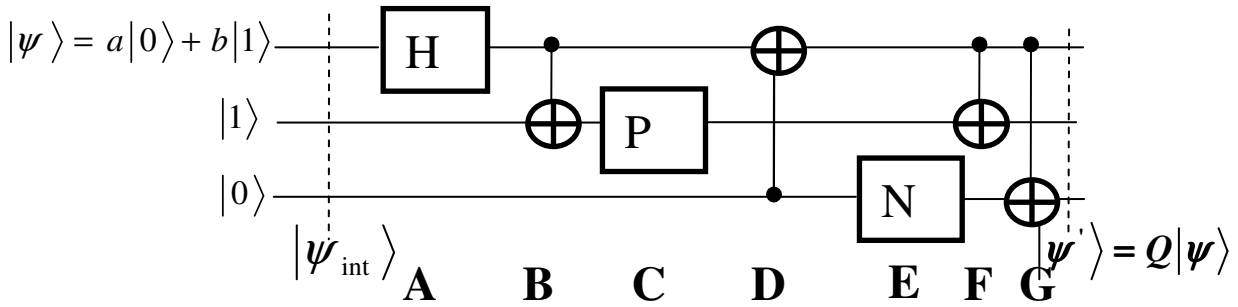
Portes Quantiques Elémentaires

Porte	Symbole	Equation	Représentation matricielle dans H
Porte Hadamard H		$\|\psi\rangle = 2^{-1/2}\left((-1)^x\|x\rangle + \|1-x\rangle\right)$	$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$
Porte Négation N		$\|\psi\rangle = \|1 + x\rangle$	$N = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$
Porte Phase P		$\|\psi\rangle = (-1)^x\|x\rangle$	$P = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$
Contrôle-Non Porte CNot		$\|\psi\rangle = \|x\rangle \otimes \|x \oplus y\rangle$ *Avec* $\otimes$: *produittensorie* $\oplus$: *XOR*	$Cnot = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}$

Classique	Quantique
Porte inverseur 	
Porte AND 	
Porte OR 	
Porte XOR 	

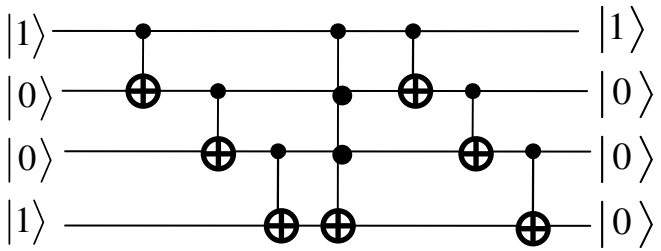
Exemples de Circuits Quantiques

- **Circuits quantiques utilisant les portes élémentaires**



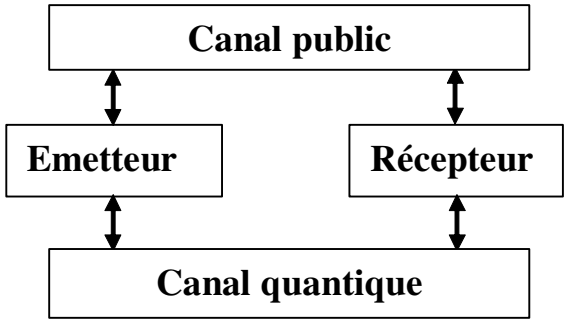
Avec $Q = G.F.E.D.C.B.A$

Circuits quantiques utilisant les portes Cnot (QBC)



PROTOCOLES D'ÉCHANGE DE CLÉS BB84

- Phase de la communication quantique



Symbole	Bit
$\|\leftrightarrow\rangle$	0
$\|\updownarrow\rangle$	1

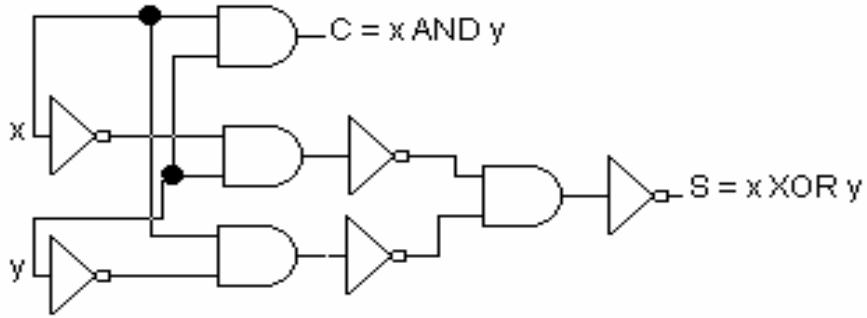
Symbole	Bit
$\|\nearrow\rangle$	0
$\|\searrow\rangle$	1

- Identification des qubits

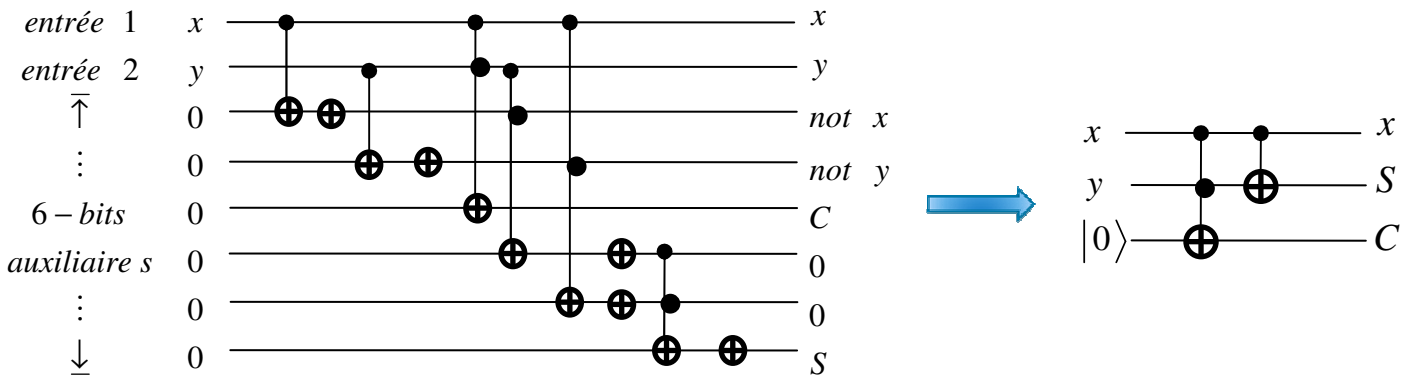
Choix d'Alice	↘↙	↕	↘↙	↔	↗↙	↔
Choix de Bob	✕	✕	✕	✛	✛	✕
Résultat de Bob	↘↙	↗↙	↘↙	↔	↕	↘↙
Compatibilité	✓		✓	✓		
Suite de bits	1		1	0		

Exemple d' illustration

- Circuit logique classique



- Circuit logique Quantique

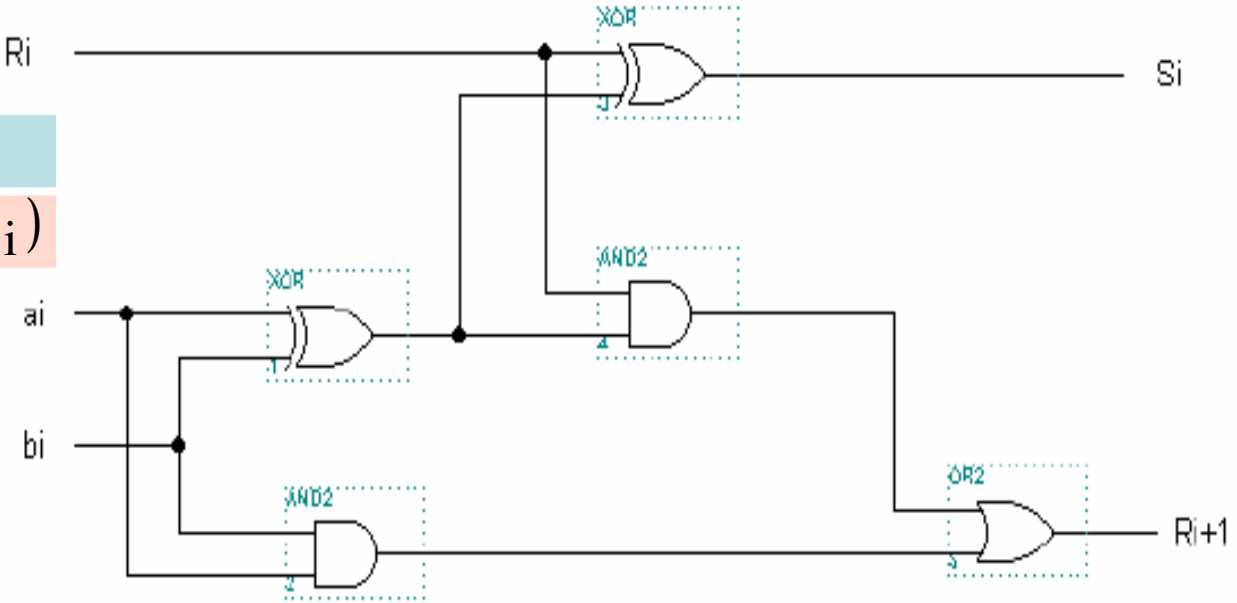


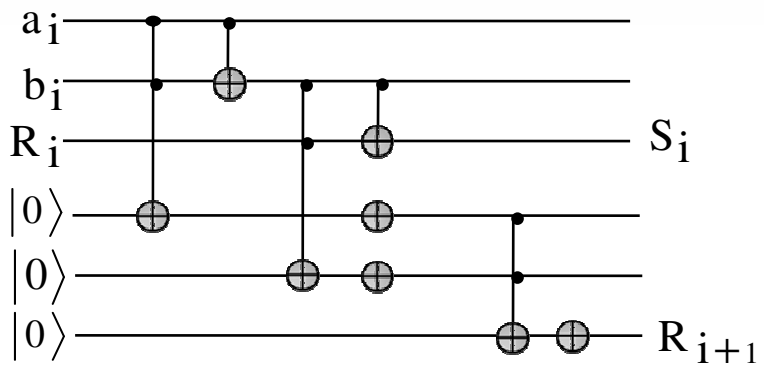
EXEMPLE DE CIRCUIT QUANTIQUE

Exemple d'illustration : Additionneur complet

$$S_i = a_i \oplus b_i \oplus R_i$$

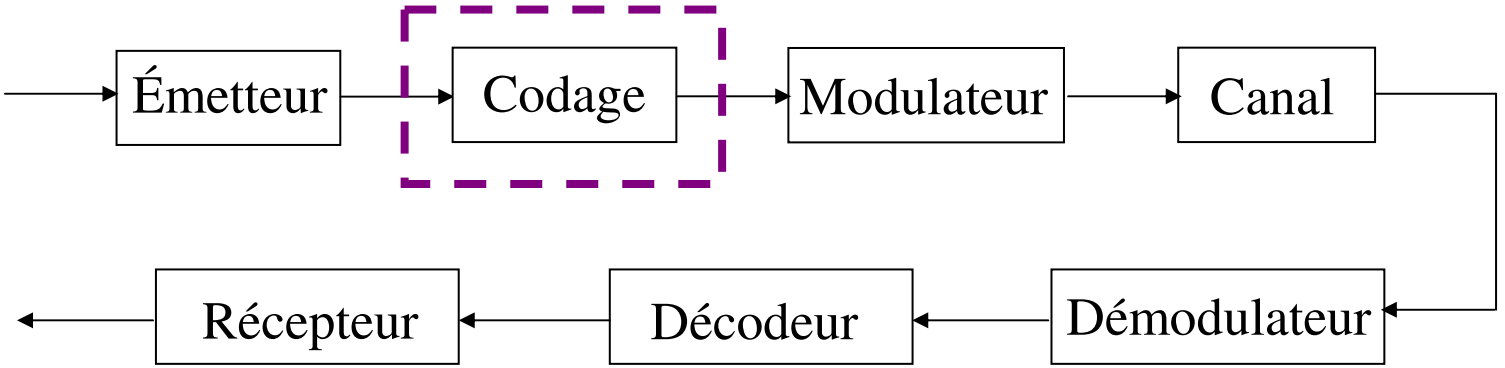
$$R_{i+1} = a_i b_i + R_i (a_i \oplus b_i)$$





Les codes correcteurs d'erreurs quantiques

- **Chaîne de transmission de données**

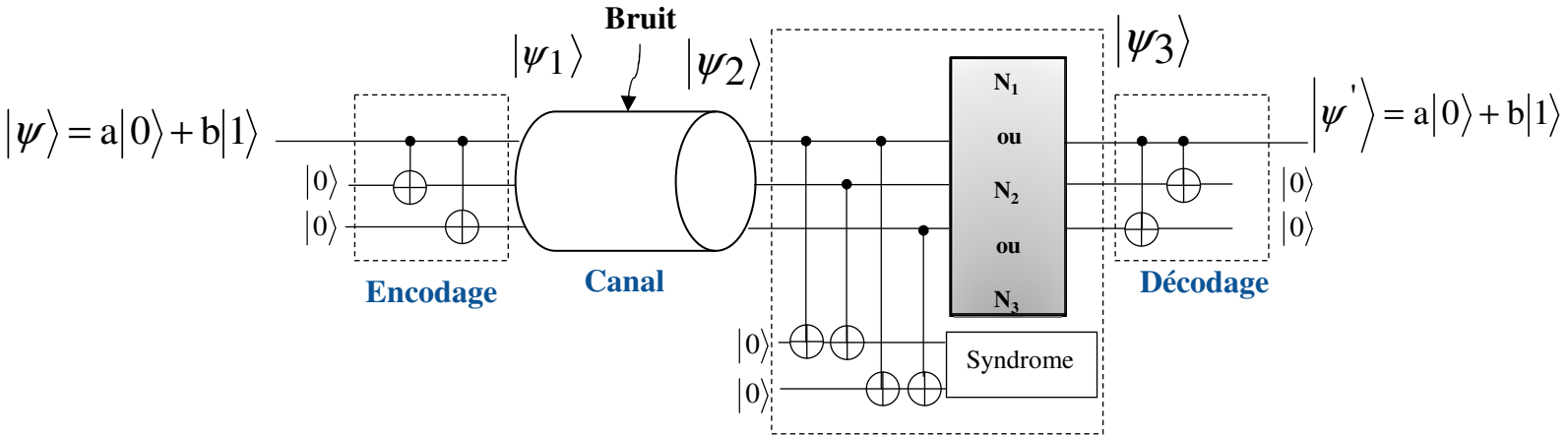


- **Définition**
  - Q[[n,k,d]]
  - Q détecte (d-1) erreurs
  - Q corrige (d-1)/2 erreurs
- **Modèles d'erreurs quantiques**

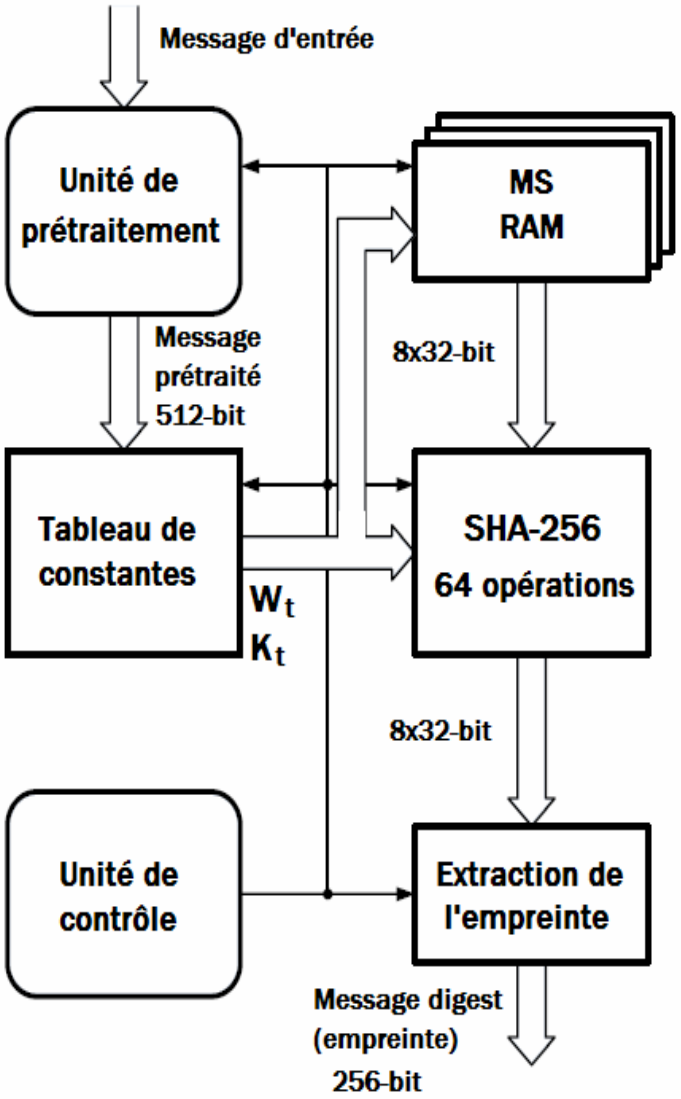
$$E_i \in \{\sigma_0 = I, \sigma_1 = N, \sigma_2 = P, \sigma_3 = NP\}$$

Merci pour votre attention

Circuit détecteur-correcteur d'une erreur d'inversion



Syndrome	correction
00	Pas d'erreur
01	Erreur sur le qubit 3
10	Erreur sur le qubit 2
11	Erreur sur le qubit 1



$$A_{t+1} = Tmp + \Sigma_0(A_t) + Maj(A_t + B_t + C_t)$$

$$B_{t+1} = A_t$$

$$C_{t+1} = B_t$$

$$D_{t+1} = C_t$$

$$E_{t+1} = Tmp + D_t$$

$$F_{t+1} = E_t$$

$$G_{t+1} = F_t$$

$$H_{t+1} = G_t$$

$$Tmp = W_{t+1} + K_{t+1} + H_t + \Sigma_1(E_t) + Ch(E_t + F_t + G_t)$$

$$H0(i) = A + H0(i-1)\ ;\ H1(i) = B + H1(i-1)\ ;$$

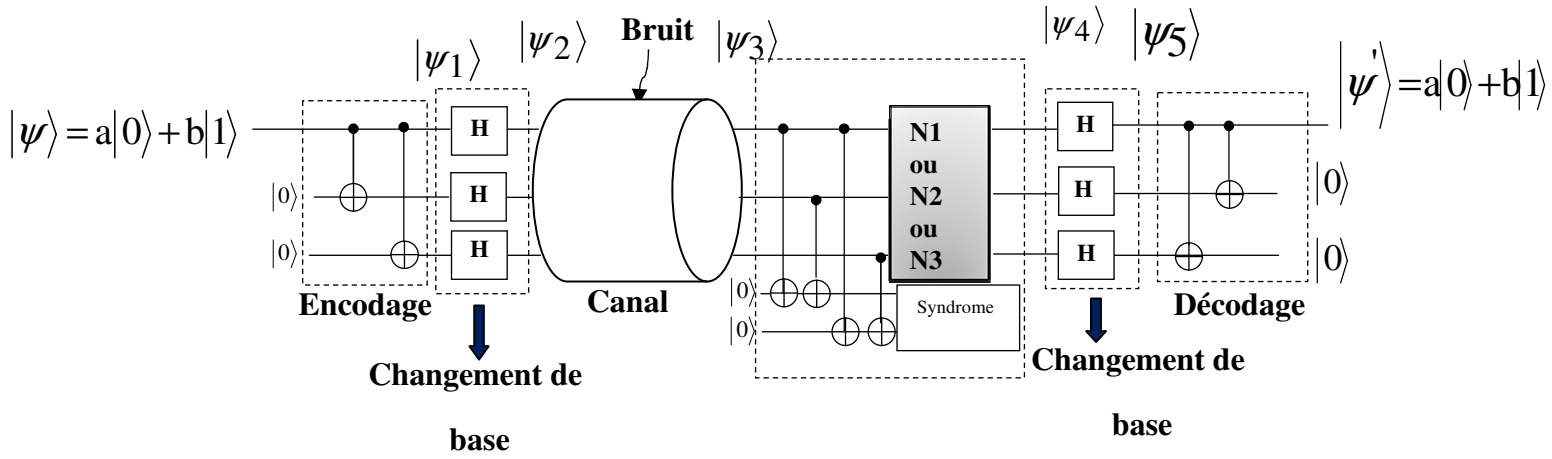
$$H2(i) = C + H2(i-1);$$

$$H3(i) = D + H3(i-1)\ ;\ H4(i) = E + H4(i-1)\ ;$$

$$H5(i) = F + H5(i-1)\ ;$$

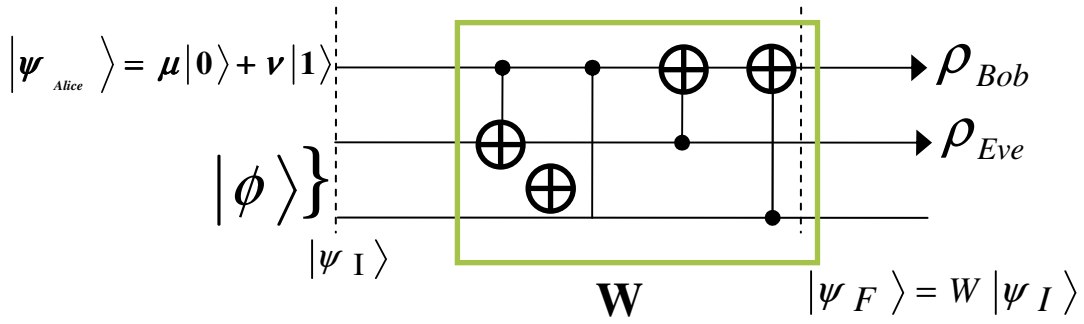
$$H6(i) = G + H6(i-1)\ ;\ H7(i) = H + H7(i-1).$$

Circuit détecteur-correcteur d'une erreur de changement de phase



- Le canal de transmission induit une erreur de changement de phase
- HPH=N
- Il suffit d'encoder l'état erroné dans un état permettant la correction d'une erreur de type N, puis d'appliquer H sur chaque qubit
- Le décodage commence par défaire H et procède de la même façon

Opérateurs densités – Calcul d'entropie



$$|\phi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

- **Avec :** $\rho = |\psi_F\rangle|\psi_F\rangle$: opérateur densité

$$\rho_{Bob} = Tr_{Bob}\{\rho\}$$

$$= \langle 00|\rho|00\rangle + \langle 01|\rho|01\rangle + \langle 10|\rho|10\rangle + \langle 11|\rho|11\rangle$$

$$= \frac{1}{2}\begin{pmatrix} |\mu|^2(|\alpha|^2 + |\delta|^2) + |\nu|^2(|\beta|^2 + |\gamma|^2) & 2\mu\nu^*\alpha\delta^* + 2\mu\nu^*\beta\gamma^* \\ 2\mu^*\nu\alpha\delta + 2\mu^*\nu\beta\gamma & |\mu|^2(|\beta|^2 + |\gamma|^2) + |\nu|^2(|\alpha|^2 + |\delta|^2) \end{pmatrix}$$

Réduction modulaire

Calculer a mod P avec $P = 2^n - c$ (Mersenne généralisé)

1) $a = a_1 2^n + a_0$
2) $a \leftarrow a_1 c + a_0$
3) $a = a_1 2^n + a_0$
4) $a \leftarrow a_1 c + a_0$