Accélerateur matériel pour la multiplication scalaire sur courbe elliptique

Nicolas Guillermin

CELAR/IRMAR

17 mars 2009

Plan

Cahier des charges

- réaliser une architecture capable de réaliser toutes les opérations de base dans GF_p ou p est un premier quelconque

Cahier des charges

- réaliser une architecture capable de réaliser toutes les opérations de base dans GF_p ou p est un premier quelconque
- Architecture générique permettant de traiter tout type d'opération

Cahier des charges

- réaliser une architecture capable de réaliser toutes les opérations de base dans GF_p ou p est un premier quelconque
- Architecture générique permettant de traiter tout type d'opération
- Utilisation pour courbes elliptiques

Cahier des charges

- réaliser une architecture capable de réaliser toutes les opérations de base dans GF_p ou p est un premier quelconque
- Architecture générique permettant de traiter tout type d'opération
- Utilisation pour courbes elliptiques
- $log(p)$ connu au moment de la génération du design

Cahier des charges

- réaliser une architecture capable de réaliser toutes les opérations de base dans GF_p ou p est un premier quelconque
- Architecture générique permettant de traiter tout type d'opération
- Utilisation pour courbes elliptiques
- $log(p)$ connu au moment de la génération du design
- p connu à la génération (pour le FPGA)

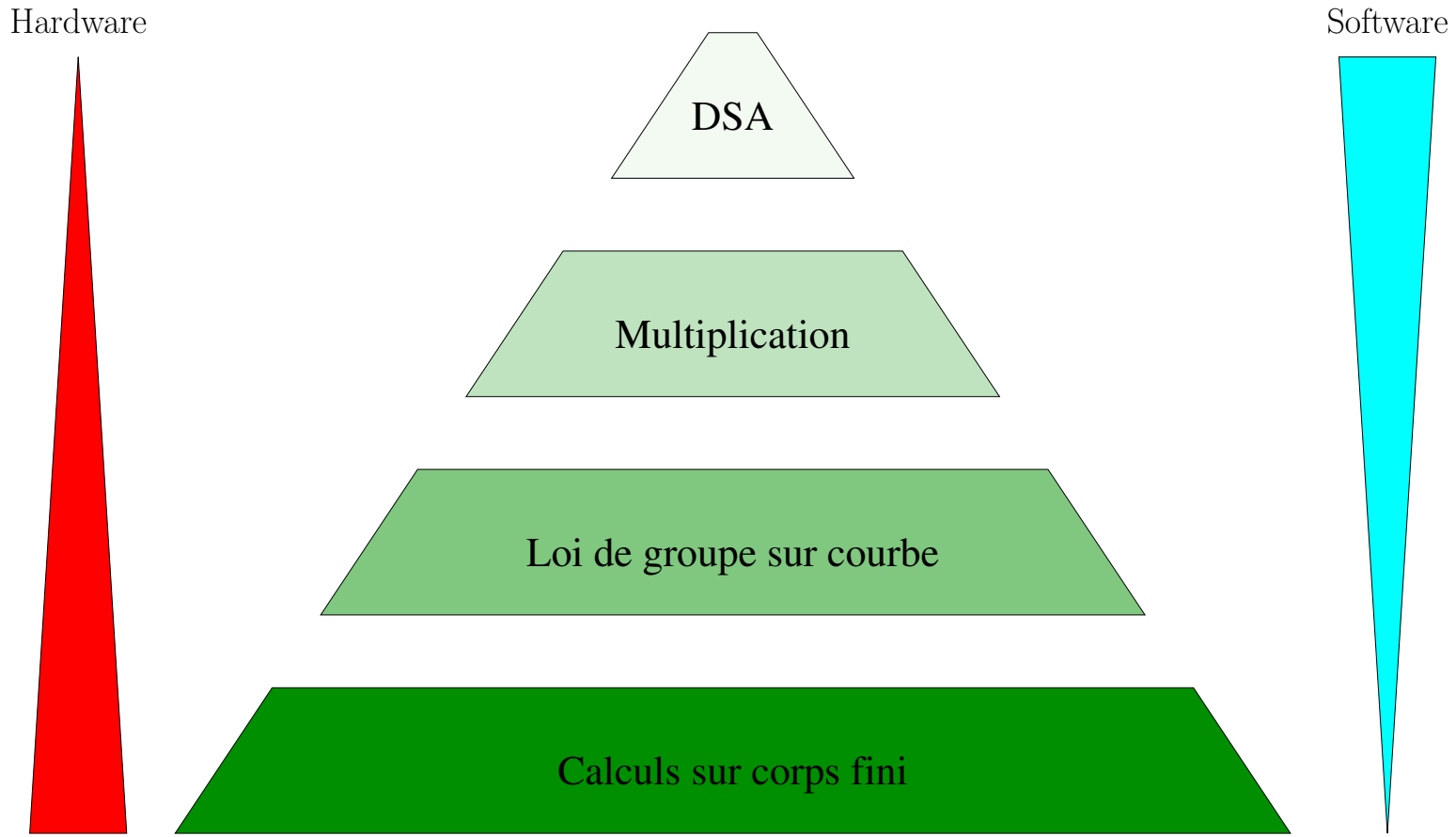
Cahier des charges

- réaliser une architecture capable de réaliser toutes les opérations de base dans GF_p ou p est un premier quelconque
- Architecture générique permettant de traiter tout type d'opération
- Utilisation pour courbes elliptiques
- $log(p)$ connu au moment de la génération du design
- p connu à la génération (pour le FPGA)
- vitesse

Cahier des charges

- réaliser une architecture capable de réaliser toutes les opérations de base dans GF_p ou p est un premier quelconque
- Architecture générique permettant de traiter tout type d'opération
- Utilisation pour courbes elliptiques
- $log(p)$ connu au moment de la génération du design
- p connu à la génération (pour le FPGA)
- vitesse
- résistance SPA (DPA)

Pile cryptographique



Multiplication scalaire et logarithme discret

Soit $(\mathbf{G}, +)$ un groupe fini et P un élément de G

Multiplication scalaire et logarithme discret

Soit $(\mathbf{G}, +)$ un groupe fini et P un élément de G

Soit $k \in \mathbb{N}$ un nombre

Multiplication scalaire et logarithme discret

Soit $(\mathbf{G}, +)$ un groupe fini et P un élément de G

Soit $k \in \mathbb{N}$ un nombre

Multiplication scalaire : $kP = \underbrace{P + P + P + ... + P}_{x \text{ fois}}$

Multiplication scalaire et logarithme discret

Soit $(\mathbf{G}, +)$ un groupe fini et P un élément de G

Soit $k \in \mathbb{N}$ un nombre

Multiplication scalaire : $kP = \underbrace{P + P + P + ... + P}_{x \text{ fois}}$

Logarithme discret : $LD(P, kP) \rightarrow k$

Multiplication scalaire et logarithme discret

Soit $(\mathbf{G}, +)$ un groupe fini et P un élément de G

Soit $k \in \mathbb{N}$ un nombre

Multiplication scalaire : $kP = \underbrace{P + P + P + ... + P}_{x \text{ fois}}$

Logarithme discret : $LD(P, kP) \rightarrow k$

Pour être utilisable en cryptographie,
le logarithme discret doit être difficile

Courbes elliptiques

Soit p un nombre premier

Courbes elliptiques

Soit p un nombre premier

Soient $(a_4, a_6) \in [0; p-1]^2$

Courbes elliptiques

Soit p un nombre premier

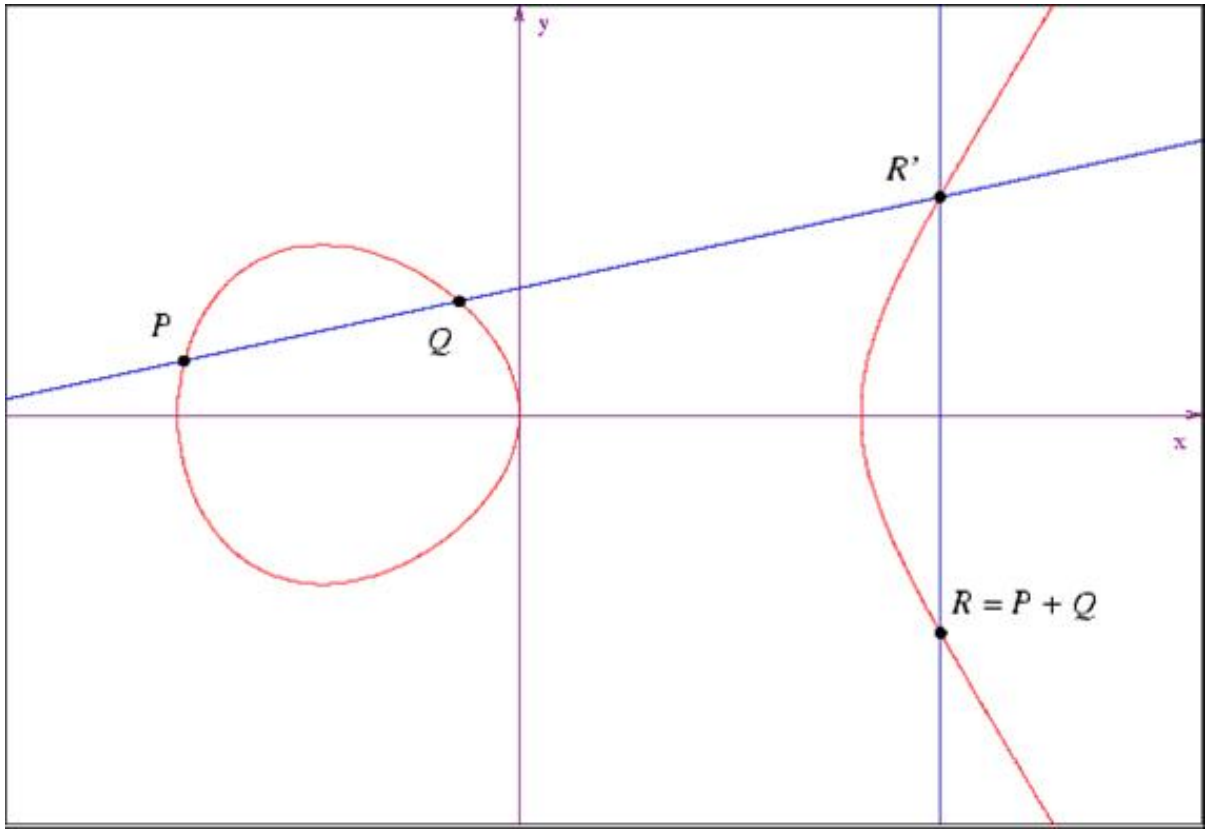
Soient $(a_4, a_6) \in [0; p-1]^2$

Une courbe elliptique $\mathbf{C}_{p,a_4,a_6}$ est l'ensemble des points $(X, Y) \in [0; p-1]^2$ tel que :

$$Y^2 = X^3 + a_4 X + a_6 \mod p \quad (1)$$

Loi de groupe

Avec le point Ω, $\mathbf{C}_{p,a_4,a_6}$ forme un groupe



Algorithme de Montgomery

Inventé par P. Montgomery en 1985

Algorithme de Montgomery

Inventé par P. Montgomery en 1985

Soit M tel que

- le calcul modulo M est facile
- la division par M est facile

Algorithme de Montgomery

Inventé par P. Montgomery en 1985

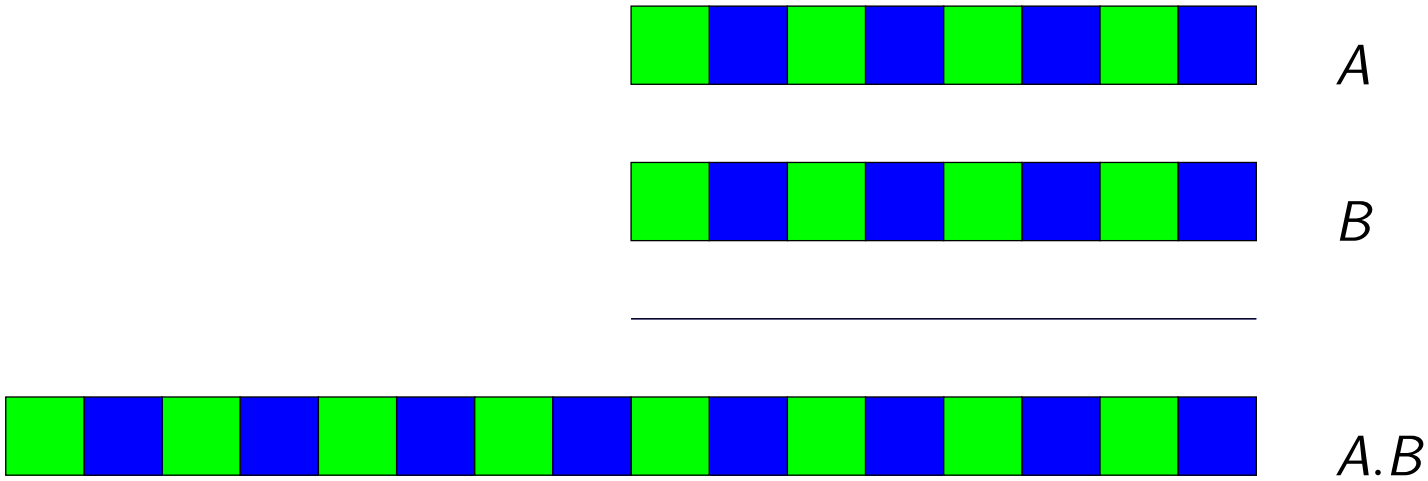
Soit M tel que

- le calcul modulo M est facile
- la division par M est facile

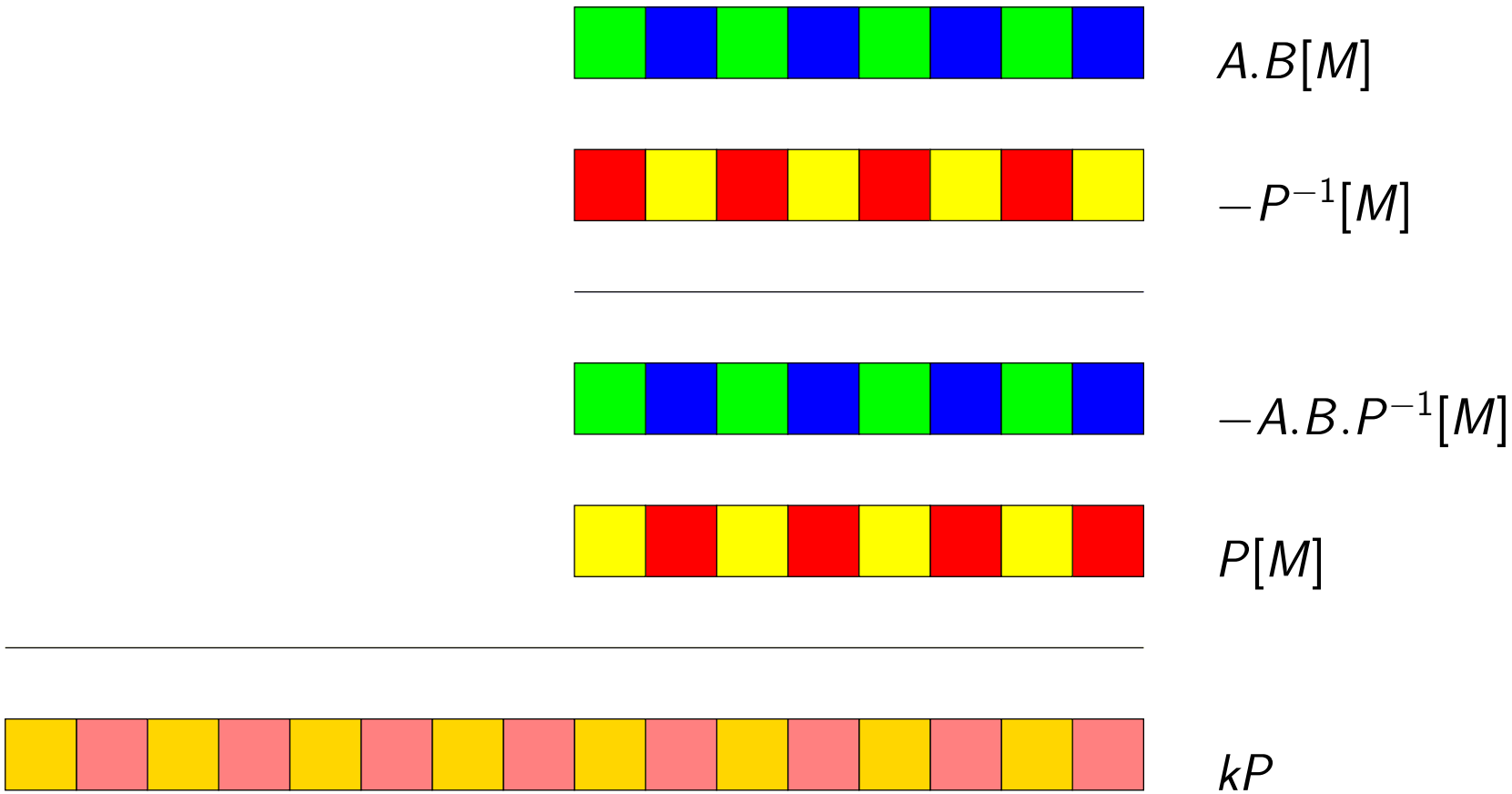
Pour tout P premier avec M, tout $A \in \mathbb{N}$, il calcule :

$$A' < A \qquad A' \equiv A.M^{-1} \mod P \qquad (2)$$

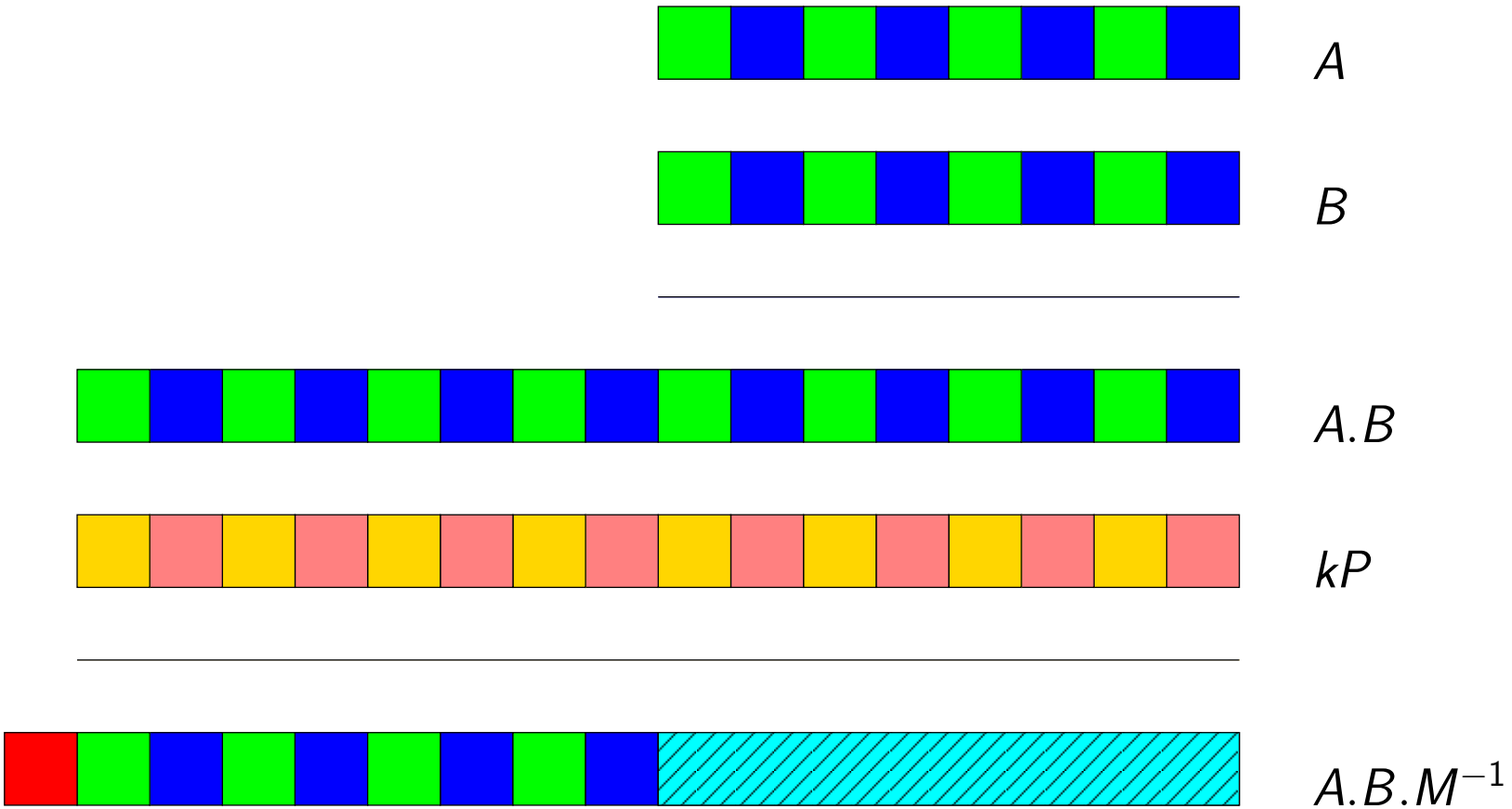
Algorithme de Montgomery(1)



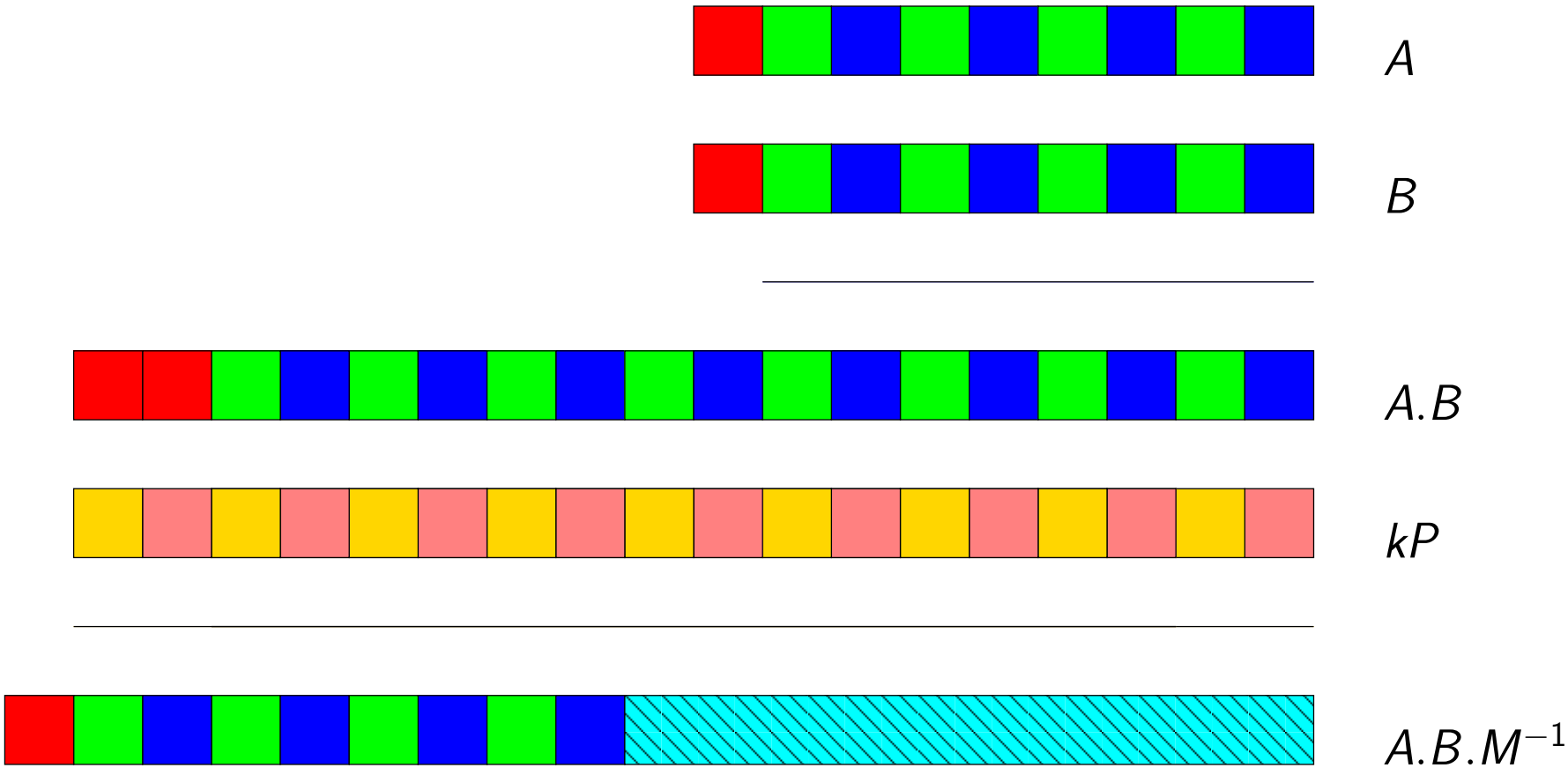
Algorithme de Montgomery(2)



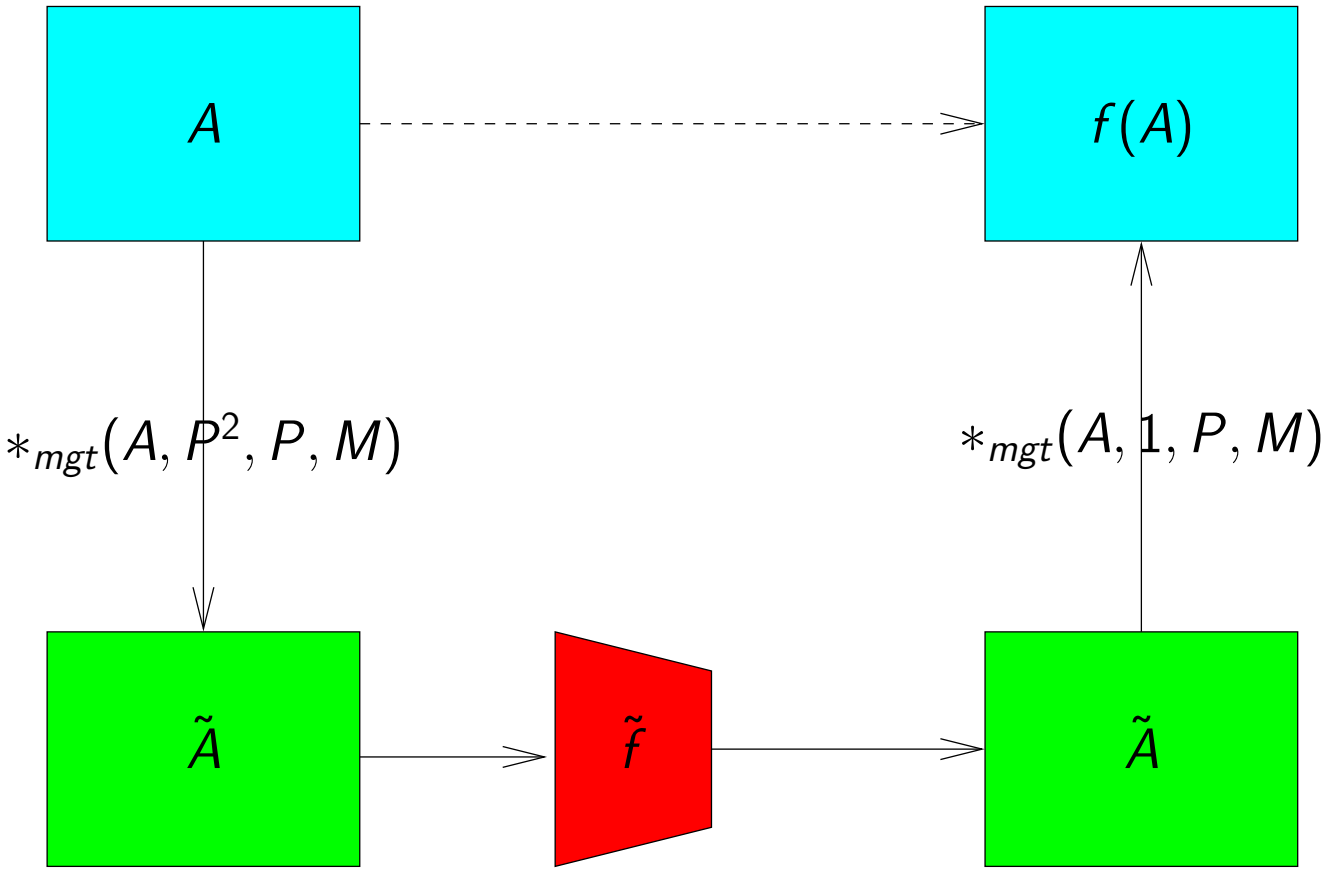
Algorithme de Montgomery(3)



Algorithme de Montgomery(4)



Algorithme de Montgomery(5)



Plan

Une alternative à l'arithmétque multiprécision

Soit $M = \prod_{i=1}^{n} m_i$ premiers entre eux

Une alternative à l'arithmétque multiprécision

Soit $M = \prod_{i=1}^{n} m_i$ premiers entre eux

La représentation RNS de $X < M$ est $\{x_1, ..., x_n\}$ tel que

$$x_i = X \mod m_i \tag{3}$$

Une alternative à l'arithmétque multiprécision

Soit $M = \prod_{i=1}^{n} m_i$ premiers entre eux

La représentation RNS de $X < M$ est $\{x_1, ..., x_n\}$ tel que

$$x_i = X \mod m_i \tag{3}$$

$$X = \sum_{i=0}^{n-1} (x_i.M_i^{-1} \mod m_i) \times M_i \mod M \tag{4}$$

$$M_i = \frac{M}{m_i} \tag{5}$$

Arithmétique dans le RNS

Arithmétique très simple dans F_M

Arithmétique dans le RNS

Arithmétique très simple dans F_M

- addition $RNS(a+b) = \{a_i + b_i\}$

Arithmétique dans le RNS

Arithmétique très simple dans F_M

- addition $RNS(a+b) = \{a_i + b_i\}$
- multiplication $RNS(a \times b) = \{a_i \times b_i\}$

Arithmétique dans le RNS

Arithmétique très simple dans F_M

- addition $RNS(a+b) = \{a_i + b_i\}$
- multiplication $RNS(a \times b) = \{a_i \times b_i\}$
- division $RNS(\frac{a}{b}) = \{\frac{a_i}{b_i}\}$

Arithmétique dans le RNS

Arithmétique très simple dans F_M

- addition $RNS(a+b) = \{a_i + b_i\}$
- multiplication $RNS(a \times b) = \{a_i \times b_i\}$
- division $RNS(\frac{a}{b}) = \{\frac{a_i}{b_i}\}$

Arithmétique dans le RNS

Arithmétique très simple dans F_M

- addition $RNS(a+b) = \{a_i + b_i\}$
- multiplication $RNS(a \times b) = \{a_i \times b_i\}$
- division $RNS(\frac{a}{b}) = \{\frac{a_i}{b_i}\}$

Perte de relation d'ordre par rapport à l'arithmétique multiprécision

Algorithme de Montgomery RNS

Algorithm 1 $Red_{Montg}(X, P, M, \tilde{M})$

Require: $X < 4P^2$ in M and $\tilde{X}$ in $\tilde{M}$, P prime, M and $\tilde{M}$ up to respectively $4P$ and $2P$

Ensure: $S \equiv X \times M^{-1}[P]$, $S < 2P$ in M and $\tilde{M}$

1: $Q \leftarrow X \times P^{-1}$ in M
2: $\tilde{Q} \leftarrow B_{ext}(Q, M, \tilde{M})$
3: $\tilde{R} \leftarrow \tilde{X} + \tilde{Q} \times \tilde{P}$ in $\tilde{M}$
4: $\tilde{S} \leftarrow \tilde{R} \times M^{-1}$ in $\tilde{M}$
5: $S \leftarrow B_{ext}(\tilde{S}, \tilde{M}, M)$
6: **return** S in M and $\tilde{S}$ in $\tilde{M}$

Avantage de Montgomery-RNS

multiplication et réduction à complexité équivalente :

- $2n^2 + n$ en multiprecision
- $2n^2 + 3n$ en RNS

multiplication en $O(n)$
reduction en $O(n^2)$

Avantage de Montgomery-RNS

multiplication et réduction à complexité équivalente :

- $2n^2 + n$ en multiprecision
- $2n^2 + 3n$ en RNS

multiplication en $O(n)$
reduction en $O(n^2)$

Idée : réduction immédiate de $\sum_{i=1}^{\alpha} A_i.B_i$

Avantage de Montgomery-RNS

multiplication et réduction à complexité équivalente :

- $2n^2 + n$ en multiprecision
- $2n^2 + 3n$ en RNS

multiplication en $O(n)$
reduction en $O(n^2)$

Idée : réduction immédiate de $\sum_{i=1}^{\alpha} A_i.B_i$

Réalisable si

- $M > 4 \times \alpha \times P$
- $\tilde{M} > 2P$

Plan

Problématique du changement de base

B_{ext} est l'opération la plus gourmande de l'algorithme

Problématique du changement de base

B_{ext} est l'opération la plus gourmande de l'algorithme

4 possibilités de réaliser cette opération

Problématique du changement de base

B_{ext} est l'opération la plus gourmande de l'algorithme

4 possibilités de réaliser cette opération

- Passage par la représentation MRS (Bajard et al)

Problématique du changement de base

B_{ext} est l'opération la plus gourmande de l'algorithme

4 possibilités de réaliser cette opération

- Passage par la représentation MRS (Bajard et al)
- Utilisation d'un module supplémentaire (Shenoy et al)

Problématique du changement de base

B_{ext} est l'opération la plus gourmande de l'algorithme

4 possibilités de réaliser cette opération

- Passage par la représentation MRS (Bajard et al)
- Utilisation d'un module supplémentaire (Shenoy et al)
- Approximation flottante (Kawamura et al)

Problématique du changement de base

B_{ext} est l'opération la plus gourmande de l'algorithme

4 possibilités de réaliser cette opération

- Passage par la représentation MRS (Bajard et al)
- Utilisation d'un module supplémentaire (Shenoy et al)
- Approximation flottante (Kawamura et al)
- Approximation rapide (Bajard et al)

Changement de base

Problématique : calculer $\tilde{x_i}$ à partir de x_i sachant

$$X = \sum_{i=0}^{n-1} (x_i.M_i^{-1} \mod m_i) \times M_i \mod M \tag{6}$$

$$\tilde{x_i} = X \mod \tilde{m}_i \tag{7}$$

Kawamura et al

Idée : calculer ε tel que

$$X = \sum_{i=0}^{n-1} (x_i.M_i^{-1} \mod m_i) \times M_i - \varepsilon M \tag{8}$$

Kawamura et al

Idée : calculer ε tel que

$$X = \sum_{i=0}^{n-1} (x_i.M_i^{-1} \mod m_i) \times M_i - \varepsilon M \tag{8}$$

$$\varepsilon = \lfloor \frac{1}{M} (\sum_{i=0}^{n-1} (x_i.M_i^{-1} \mod m_i) \times M_i) \rfloor \tag{9}$$

Kawamura et al

Idée : calculer ε tel que

$$X = \sum_{i=0}^{n-1} (x_i.M_i^{-1} \mod m_i) \times M_i - \varepsilon M \tag{8}$$

$$\varepsilon = \lfloor \frac{1}{M} (\sum_{i=0}^{n-1} (x_i.M_i^{-1} \mod m_i) \times M_i) \rfloor \tag{9}$$

$$\varepsilon = \lfloor (\sum_{i=0}^{n-1} \frac{\xi_i}{m_i}) \rfloor \tag{10}$$

Algorithme $B_{ext}(M, \tilde{M})(1)$

Algorithm 2 $B_{ext}(M, \tilde{M})$

Require: X in M

Ensure: $X, X + M$ or $X - M$ in $\tilde{M}$

1: $\xi_i \leftarrow x_i \times M_i^{-1} \mod m_i$
2: $\tilde{x_i} \leftarrow 0$
3: $\varepsilon \leftarrow 0.0$
4: **for** j from 0 to $n-1$ **do**
5: $\quad \tilde{x_i} \leftarrow \tilde{x_i} + \xi_j \times M_j mod \tilde{m}_i$
6: $\quad \varepsilon \leftarrow \varepsilon + \frac{\xi_j}{m_j}$
7: **end for**
8: $\tilde{x_i} \leftarrow \tilde{x_i} - \varepsilon \times M \mod \tilde{m}_i$
9: **return** $\tilde{x_i}$

Algorithme $B_{ext}(M, \tilde{M})(2)$

Idée : utiliser des bases de pseudo-mersenne

$$m_i = 2^{radix} - \mu_i \tag{11}$$

$$\mu_i < 2^{\frac{radix}{2}} \tag{12}$$

et approximer $\frac{\xi_i}{m_i}$ par $\frac{\xi_i}{2^{radix}}$

Algorithme $B_{ext}(M, \tilde{M})(3)$

Algorithm 3 $B_{ext}(M, \tilde{M})$

Require: X in M

Ensure: X,or $X + M$ in $\tilde{M}$

1: $\xi_i \leftarrow x_i \times M_i^{-1} \mod m_i$
2: $\tilde{x_i} \leftarrow 0$
3: $\varepsilon \leftarrow 0.0$
4: **for** j from 0 to $n-1$ **do**
5: $\quad \tilde{x_i} \leftarrow \tilde{x_i} + \xi_j \times M_j mod \tilde{m}_i$
6: $\quad \varepsilon \leftarrow \varepsilon + \frac{\xi_j}{2^{radix}}$
7: **end for**
8: $\tilde{x_i} \leftarrow \tilde{x_i} - \varepsilon \times M \mod \tilde{m}_i$
9: **return** $\tilde{x_i}$

Algorithme de Montgomery RNS

Algorithm 4 $Red_{Montg}(X, P, M, \tilde{M})$

Require: $X < 4P^2$ in M and $\tilde{M}$, P prime, M and $\tilde{M}$ up to respectively $4P$ and $2P$

Ensure: $S \equiv X \times M^{-1}[P]$, $S < 2P$ in M and $\tilde{M}$

1: $Q \leftarrow X \times P^{-1}$ in M
2: $\tilde{Q} \leftarrow B_{ext}(Q, M, \tilde{M}) + M$
3: $\tilde{R} \leftarrow X + (\tilde{Q} + M) \times \tilde{P}$ in $\tilde{M}$
4: $\tilde{S} \leftarrow (R + \tilde{P}M) \times M^{-1}$ in $\tilde{M}$
5: $S \leftarrow B_{ext}(\tilde{S} + \tilde{P}, \tilde{M}, M) + \tilde{M}$
6: **return** $\tilde{S} + \tilde{P}$ in $\tilde{M}$ and $S + P + \tilde{M}$ in M

Changement de base

Erreur sur premier changement de base : pas de conséquence si

1. $X \in [0; 3P[$

Changement de base

Erreur sur premier changement de base : pas de conséquence si

1. $X \in [0; 3P[$
2. $M > 9kP$

Changement de base

Erreur sur premier changement de base : pas de conséquence si

1. $X \in [0; 3P[$
2. $M > 9kP$
3. $\tilde{M} > 3P$

Changement de base

Erreur sur premier changement de base : pas de conséquence si

1. $X \in [0; 3P[$
2. $M > 9kP$
3. $\tilde{M} > 3P$

Pas d'erreur sur le deuxième changement

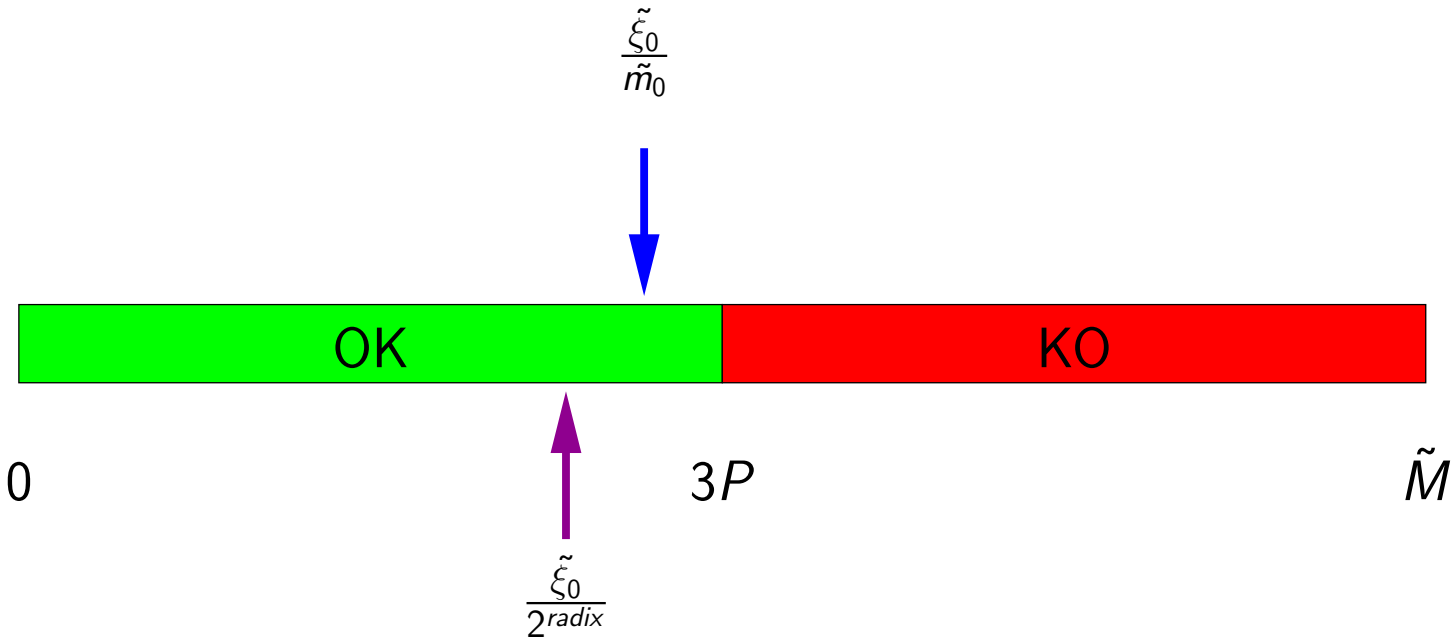
Deuxième changement de base

Idée : augmenter $\tilde{M}$ de $3P$ à $6P$

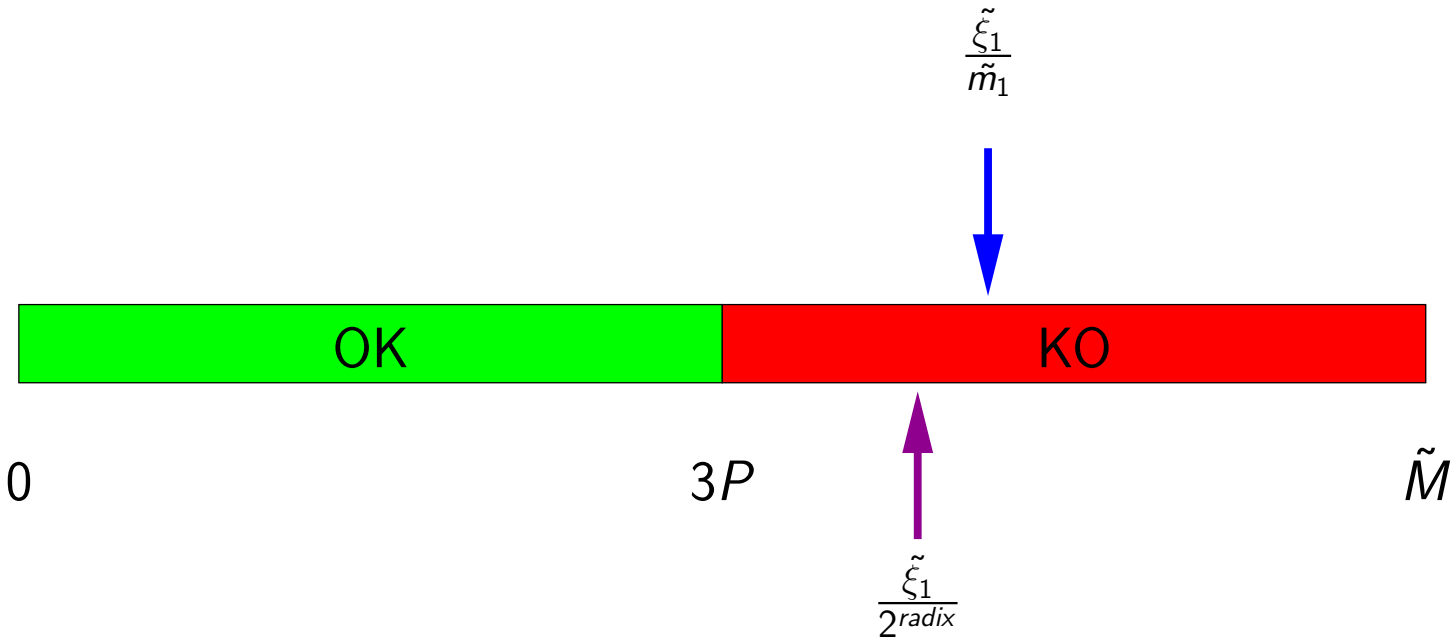
Evaluer la valeur suivante :

$$\frac{X}{\tilde{M}} = \sum_{i=0}^{n-1}(\frac{\tilde{\xi}_i}{\tilde{m}_i}) - \lfloor\sum_{i=0}^{n-1}(\frac{\tilde{\xi}_i}{\tilde{m}_i})\rfloor < \frac{1}{2} \tag{13}$$

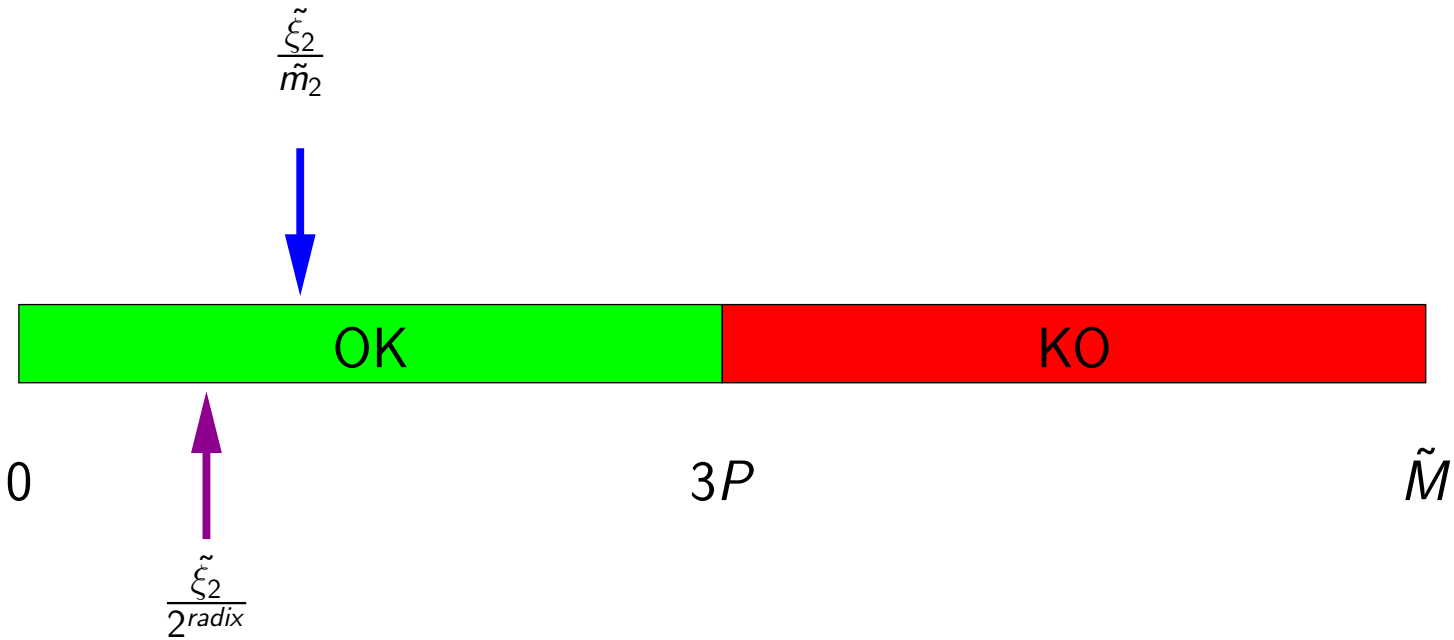
Deuxième changement de base



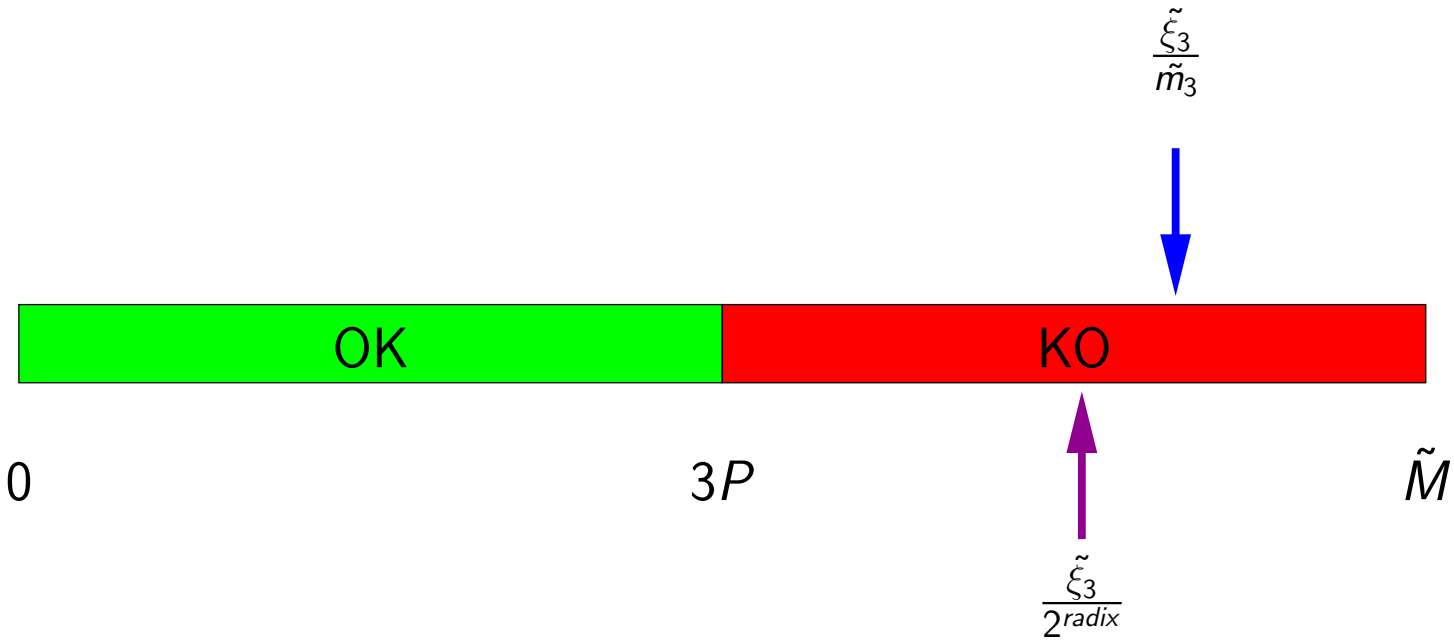
Deuxième changement de base



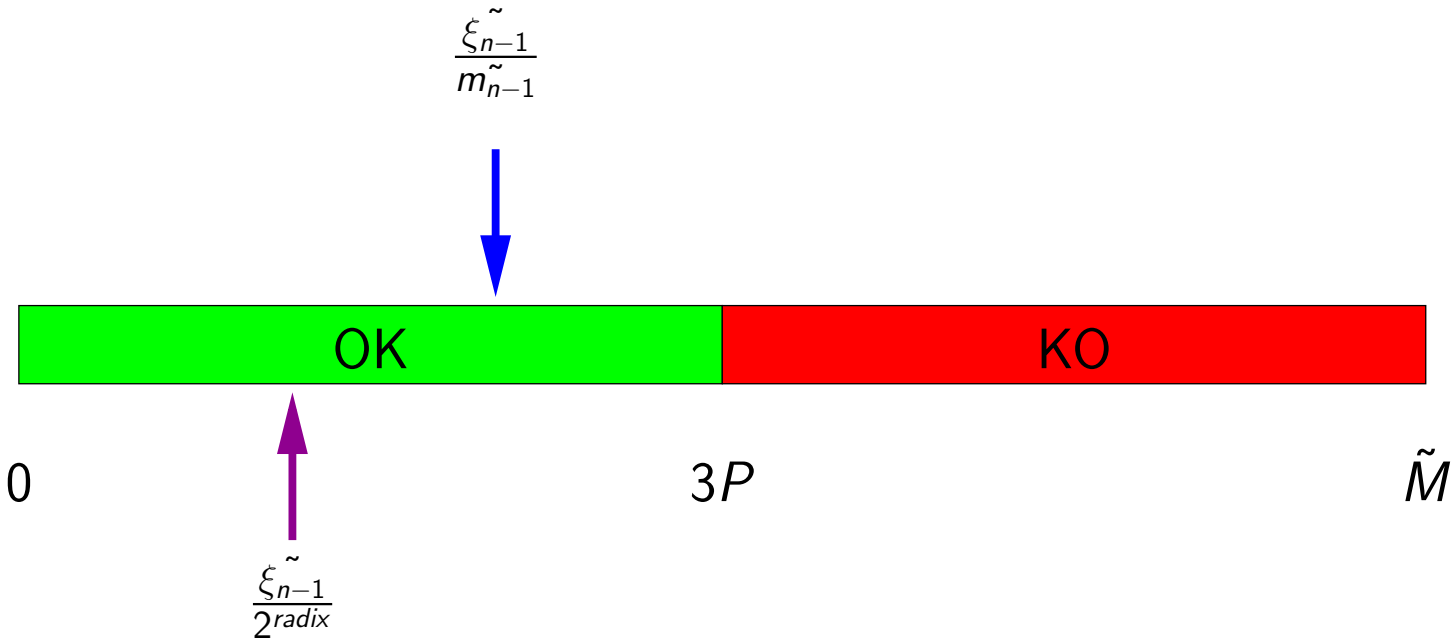
Deuxième changement de base



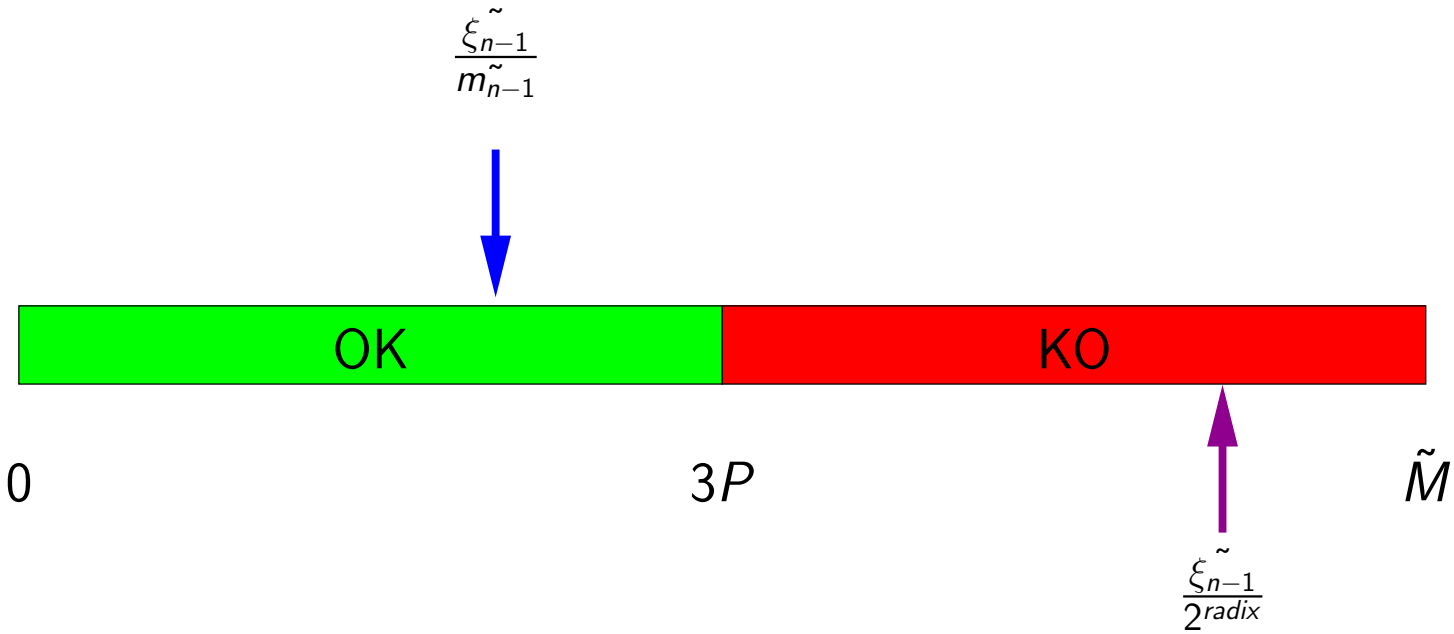
Deuxième changement de base



Quand ça se passe bien



Quand ça se passe mal



Comparatif

Algorithme	baj	she	kaw	baj2
Complexité	$\frac{3}{2}n^2$	$n^2 + O(n)$	$n^2 + o(n)$	n^2
Représentant	$2P$	$2P$	$4P$	$(n+1)P$
$B_{ext}(\tilde{M}, M)$	oui	oui	oui	non
parallélisation	-	$+$	$++$	$+++$

Plan

- Complexité plus importante ($2n^2 + 3n$ par réduction)
- Opération de base plus complexe ($a \times b \mod m$)

- Complexité plus importante ($2n^2 + 3n$ par réduction)
- Opération de base plus complexe ($a \times b \mod m$)

Mais

- Le RNS permet de porter toute la complexité sur la réduction modulaire

- Complexité plus importante ($2n^2 + 3n$ par réduction)
- Opération de base plus complexe ($a \times b \mod m$)

Mais

- Le RNS permet de porter toute la complexité sur la réduction modulaire
- Recherche de pattern de type $A \times B + C \times D$

Résistance SPA

2 solutions :

Résistance SPA

2 solutions :

1. Utiliser des formules d'addition et doublement unifiée
2. Utiliser l'algorithme *Montg – ladder*

$Montg - ladder$

Algorithm 5 $Montg - ladder(k, P, \mathbf{C}_{p,a_4,a_6})$

1: $R \leftarrow \Omega$
2: $S \leftarrow P$
3: **for** i from $size(k)$ to 0 **do**
4: **if** $(k_i = 0)$ **then**
5: $R \leftarrow 2 \times R$
6: $S \leftarrow R + S$
7: **else**
8: $S \leftarrow 2 \times S$
9: $R \leftarrow R + S$
10: **end if**
11: **end for**
12: **return** R

Representation de points

- Utilisation des coordonnées projectives (X_P, Y_P, Z_P)

$$x_P = \frac{X_P}{Z_P} \tag{14}$$

$$y_P = \frac{X_P}{Z_P} \tag{15}$$

Representation de points

- Utilisation des coordonnées projectives (X_P, Y_P, Z_P)

$$x_P = \frac{X_P}{Z_P} \tag{14}$$

$$y_P = \frac{X_P}{Z_P} \tag{15}$$

- Calcul sur les surfaces de Kummer

Representation de points

- Utilisation des coordonnées projectives (X_P, Y_P, Z_P)

$$x_P = \frac{X_P}{Z_P} \tag{14}$$

$$y_P = \frac{X_P}{Z_P} \tag{15}$$

- Calcul sur les surfaces de Kummer

  y_{kP}reconstruit à la fin du calcul

$$y_{kP} = (2a_6+(a_4+x_Px_{kP})(x_P+x_{kP})-x_{(k+1)P}(x_P-x_{kP})^2)(2y_P)^{-1} \tag{16}$$

Representation de points

- Utilisation des coordonnées projectives (X_P, Y_P, Z_P)

$$x_P = \frac{X_P}{Z_P} \tag{14}$$

$$y_P = \frac{X_P}{Z_P} \tag{15}$$

- Calcul sur les surfaces de Kummer

  y_{kP}reconstruit à la fin du calcul

$$y_{kP} = (2a_6 + (a_4 + x_P x_{kP})(x_P + x_{kP}) - x_{(k+1)P}(x_P - x_{kP})^2)(2y_P)^{-1} \tag{16}$$

  1 inversions à la fin du calcul de kP (hors de ce contexte)

Formules d'addition

Etape	Comp	réd
précalculs	$A = Z_P X_Q + X_P Z_Q$, $B = 2X_P X_Q$	2
$P + Q$	$C = 2Z_P Z_Q$, $D = a_4 A + a6 C$	2
Z_{P+Q}	$A^2 - BC$	1
X_{P+Q}	$BA + CD + 2xZ_{P+Q}$	1
total		6

Formules de doublement

Etape	Comp	réd
précalculs	$A' = 2X_P Z_P$, $B' = X_P^2$, $C' = Z_P^2$	3
$2P$	$D' = -4bA$, $E' = aA$	2
Z_{2P}	$B'D' + (C' - E')^2$	1
X_{2P}	$2B'(C' + E') - A'D'$	1
total		7

Conclusion

- RNS adapté pour réaliser des *Montg* – *ladder*

Conclusion

- RNS adapté pour réaliser des $Montg - ladder$
- Coût par bit de l'exposant $13 \times 2 = 26n^2$ au lieu de $19 + 13 = 32n^2$

Conclusion

- RNS adapté pour réaliser des $Montg - ladder$
- Coût par bit de l'exposant $13 \times 2 = 26n^2$ au lieu de $19 + 13 = 32n^2$
- résistance native SPA $\Rightarrow$ anti-DPA possible

Conclusion

- RNS adapté pour réaliser des $Montg - ladder$
- Coût par bit de l'exposant $13 \times 2 = 26n^2$ au lieu de $19 + 13 = 32n^2$
- résistance native SPA $\Rightarrow$ anti-DPA possible
- Aucune restriction sur le choix de la courbe

Plan

FPGA cible

Choix d'un Altera de la famille Stratix 2

FPGA cible

Choix d'un Altera de la famille Stratix 2

FPGA : Matrice d'interconnection d'éléments de base qui sont

- fonctions logiques 4 vers 1 programmables (ALUT)

FPGA cible

Choix d'un Altera de la famille Stratix 2

FPGA : Matrice d'interconnection d'éléments de base qui sont

- fonctions logiques 4 vers 1 programmables (ALUT)
- blocs DSP 9×9, 18×18 et 36×36

FPGA cible

Choix d'un Altera de la famille Stratix 2

FPGA : Matrice d'interconnection d'éléments de base qui sont

- fonctions logiques 4 vers 1 programmables (ALUT)
- blocs DSP 9×9, 18×18 et 36×36
- blocs de mémoire.

FPGA cible

Choix d'un Altera de la famille Stratix 2

FPGA : Matrice d'interconnection d'éléments de base qui sont

- fonctions logiques 4 vers 1 programmables (ALUT)
- blocs DSP 9×9, 18×18 et 36×36
- blocs de mémoire.
- Blocs d'IO

FPGA cible

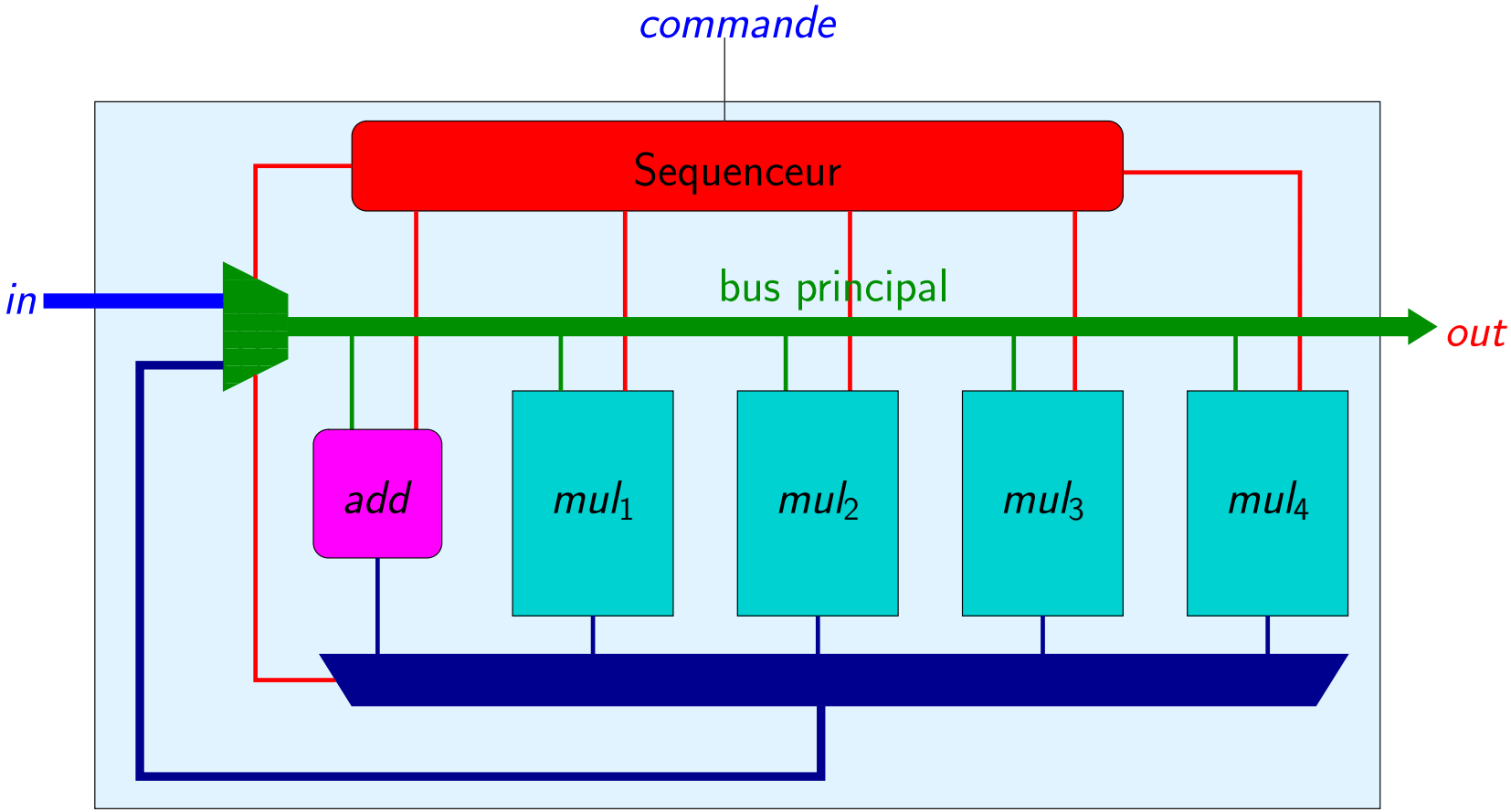
Choix d'un Altera de la famille Stratix 2

FPGA : Matrice d'interconnection d'éléments de base qui sont

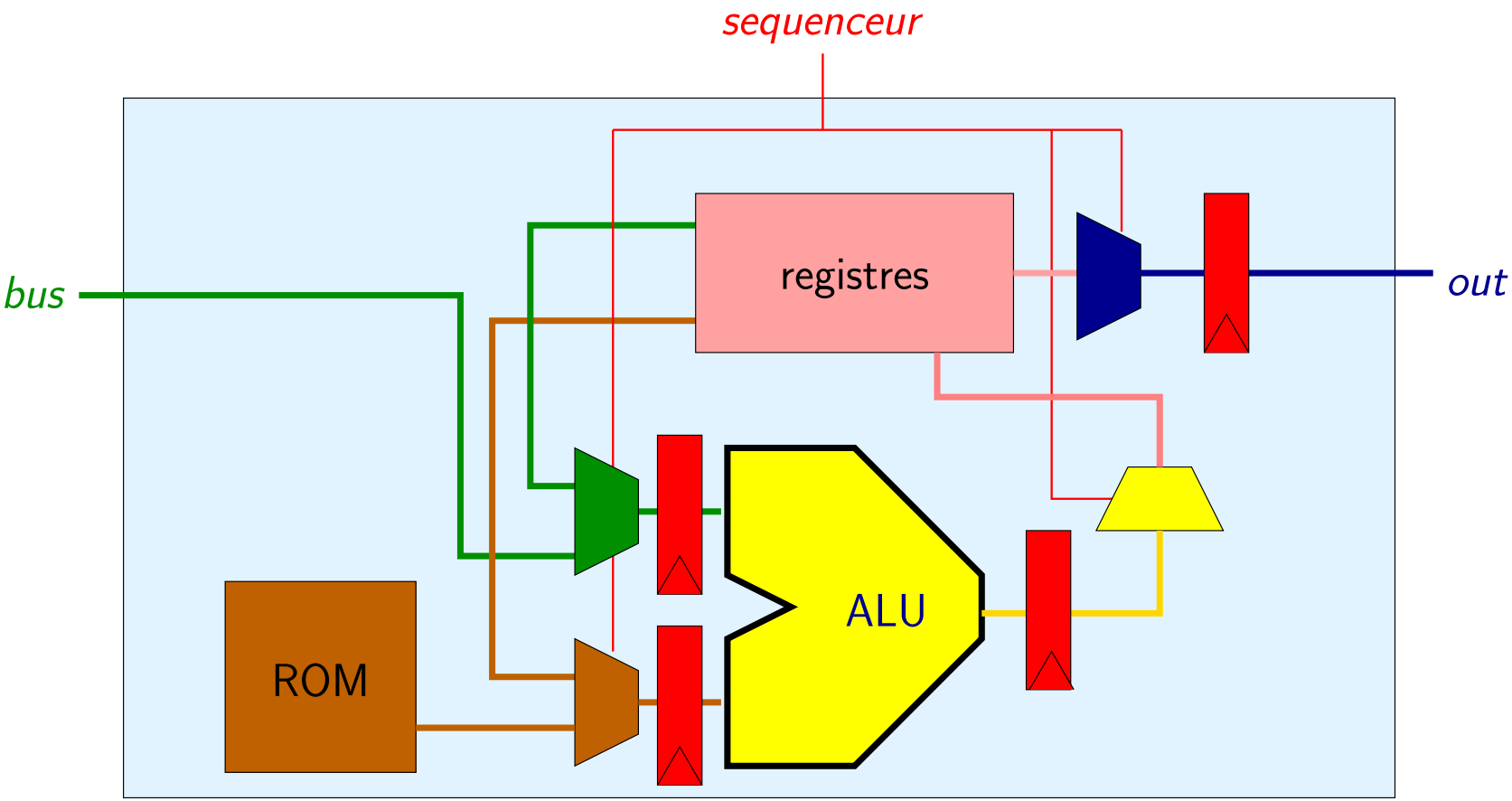
- fonctions logiques 4 vers 1 programmables (ALUT)
- blocs DSP 9×9, 18×18 et 36×36
- blocs de mémoire.
- Blocs d'IO

L'utilisation est synchrone (cadencée par une horloge)

Vue d'ensemble



Canaux de multiplication



Nombre de registres

Les données contenues (hors courbe) dans les registres sont :

- $X_{kP}, Z_{kP}, X_{(k+1)P}, Z_{(k+1)P}$

Nombre de registres

Les données contenues (hors courbe) dans les registres sont :

- $X_{kP}, Z_{kP}, X_{(k+1)P}, Z_{(k+1)P}$
- A, B, C , D pour la somme de points

Nombre de registres

Les données contenues (hors courbe) dans les registres sont :

- $X_{kP}, Z_{kP}, X_{(k+1)P}, Z_{(k+1)P}$
- A, B, C , D pour la somme de points
- A' , B' , C' , D' , E' pour le doublement

Nombre de registres

Les données contenues (hors courbe) dans les registres sont :

- $X_{kP}, Z_{kP}, X_{(k+1)P}, Z_{(k+1)P}$
- A, B, C , D pour la somme de points
- A' , B' , C' , D' , E' pour le doublement
- x_P qui doit toujours être disponible

Soit au maximum 8 variables en même temps. 16 registres par canaux suffisent

Unité arithmétique et logique

Sur chaque canal, à chaque tour

- Calcul de $reg_1 \times reg_2 \mod r$

Unité arithmétique et logique

Sur chaque canal, à chaque tour

- Calcul de $reg_1 \times reg_2 \mod r$
- r choisi parmi 2 : m_i ou $\tilde{m}_i$

Unité arithmétique et logique

Sur chaque canal, à chaque tour

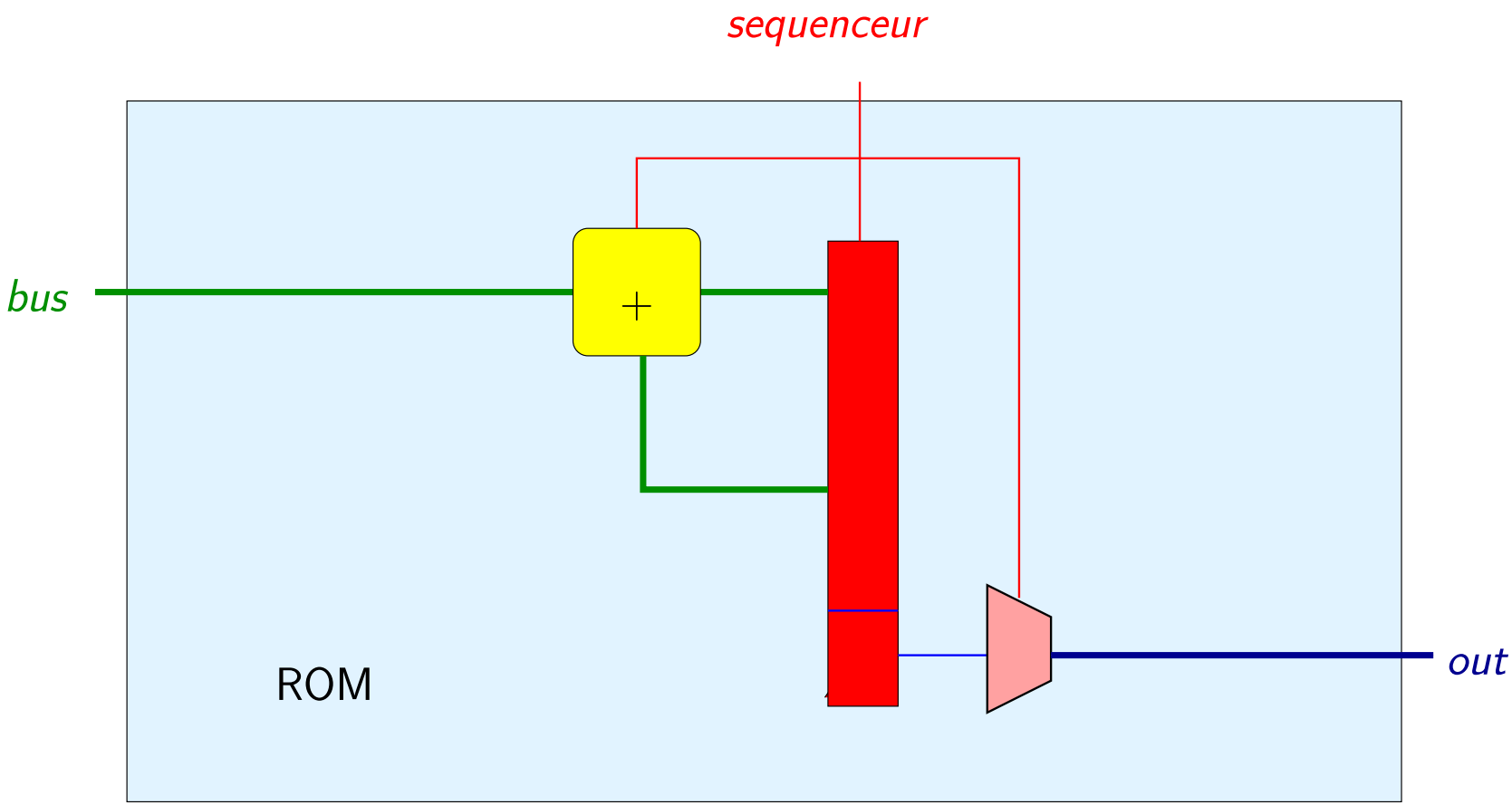
- Calcul de $reg_1 \times reg_2 \mod r$
- r choisi parmi 2 : m_i ou $\tilde{m}_i$
- possibilité d'accumuler le résultat en 1 cycle

Unité arithmétique et logique

Sur chaque canal, à chaque tour

- Calcul de $reg_1 \times reg_2 \mod r$
- r choisi parmi 2 : m_i ou $\tilde{m}_i$
- possibilité d'accumuler le résultat en 1 cycle
- utilisation des multiplieurs 18×18 des FPGA

additionneur



Considérations de performance

- Les vitesses maximales définies par le routage des variables

Considérations de performance

- Les vitesses maximales définies par le routage des variables

courbe	canal	canaux	vitesse
192	18	11	179 MHz
192	36	6	175 MHz
512	18	29	130 MHz
512	36	15	165 MHz

Considérations de performance

- Les vitesses maximales définies par le routage des variables

courbe	canal	canaux	vitesse
192	18	11	179 MHz
192	36	6	175 MHz
512	18	29	130 MHz
512	36	15	165 MHz

- Conclusion : les chemins critiques sont dans les multiplieurs.

Profondeur des pipelines

Plus le pipeline est profond :

Profondeur des pipelines

Plus le pipeline est profond :

- plus la fréquence est importante

Profondeur des pipelines

Plus le pipeline est profond :

- plus la fréquence est importante
- plus le taux d'inutilisation est haut

Profondeur des pipelines

Plus le pipeline est profond :

- plus la fréquence est importante
- plus le taux d'inutilisation est haut

Idée : paralléliser les opérations dans chaque tour

Profondeur des pipelines

Plus le pipeline est profond :

- plus la fréquence est importante
- plus le taux d'inutilisation est haut

Idée : paralléliser les opérations dans chaque tour

- augmente le taux d'occupation du pipeline

Profondeur des pipelines

Plus le pipeline est profond :

- plus la fréquence est importante
- plus le taux d'inutilisation est haut

Idée : paralléliser les opérations dans chaque tour

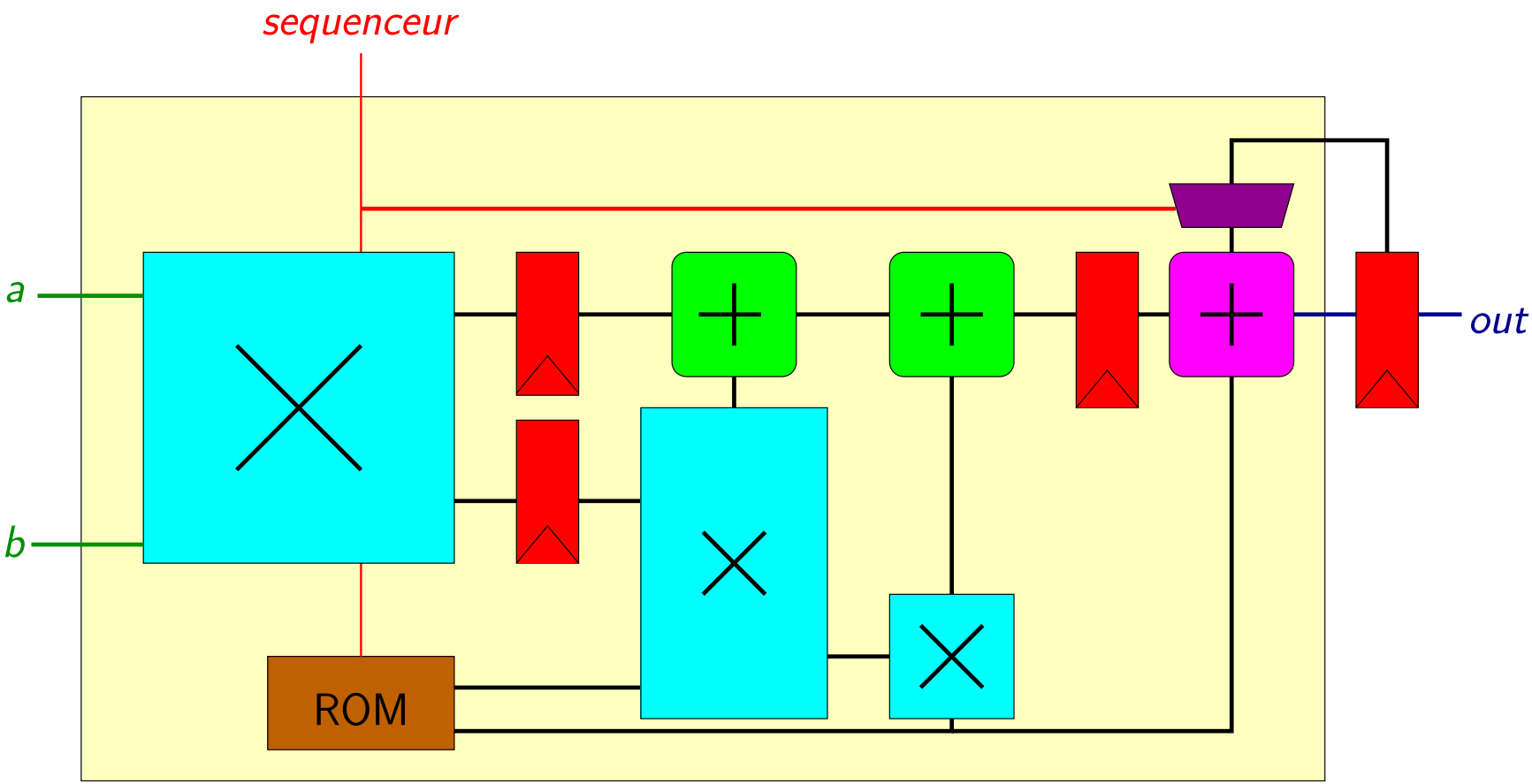
- augmente le taux d'occupation du pipeline
- augmente le nombre de registres nécessaires
- pas toujours possible

Parallélisation proposée

Respect des 8 variables locales en mémoire

Etape 1	Etape 2	Etape 3	Etape 4	Etape 5
A B C	D Z_{P+Q}	X_{P+Q} A' B' C'	D' E'	X_{2P} Z_{2P}
3	2	4	2	2

proposition d'ALU



Résultats provisoires : multiplieurs 18×18

Pour un pipeline à 3 étages et 16 registres
Sans réduction finale

Résultats provisoires : multiplieurs 18 × 18

Pour un pipeline à 3 étages et 16 registres
Sans réduction finale

courbe	freq (MHz)	taille (ALUT)	DSP	vitesse (ms)	taux
160	121	4922	50	0.58	83%
192	119	5339	55	0.75	84%
256	103	8100	75	1.38	87%
512	105	16505	124	4.50	92%

Résultats provisoires : multiplieurs 36×36

Pour un pipeline à 3 étages et 16 registres

Sans réduction finale

Résultats provisoires : multiplieurs 36 × 36

Pour un pipeline à 3 étages et 16 registres

Sans réduction finale

courbe	freq (MHz)	taille (ALUT)	DSP	vitesse (ms)	taux
160	102	4689	85	0.51	76%
192	111	5758	96	0.60	78%
256	105	7885	128	0.93	81%
512	87	23734	255	3.3	87%

Comparaison à la concurrence

A faire...

Comparaison à la concurrence

A faire... mais c'est prometteur

Evidemment, le sponsor

Evidemment, le sponsor