Représentation des Nombres et Algorithmes

Implémentation Complètement RNS du Schéma de Chiffrement de Fan et Vercauteren

Vincent Zucca

Année 2016-2017

Rappels sur le RNS et FV

Théorème des Restes Chinois

Si $q \in \mathbb{Z}$ est produit d'entiers **premiers entre eux** deux à deux $\{q_1, \ldots, q_k\}$ formant la "base RNS", on a l'isomorphisme d'anneaux :

$$\mathbb{Z}/q\mathbb{Z} \cong \mathbb{Z}/q_1\mathbb{Z} \times \ldots \times \mathbb{Z}/q_k\mathbb{Z}$$

Residue Number Systems

- "Gros" $x \in [0, q) \cap \mathbb{Z} \leftrightarrow k$ **petits résidus** $(x \bmod q_1, \ldots, x \bmod q_k)$.
- Arithmétique en **parallèle**, **sans retenue** $(+, -, \times, \div)$ sur les résidues.
- Système de représentation **non positionnel**.

$\rightarrow$ la représentation RNS d'un nombre x dépend de la base choisie.

Comment convertir les résidues de x d'un base $\mathcal{B}$ vers une base $\mathcal{B}'$?

Rappels sur le RNS et FV

Pour convertir les résidus d'un nombre x d'une base $\mathcal{B}$ vers une base $\mathcal{B}'$ il est nécessaire de "reconstruire" x depuis ses résidus dans la base $\mathcal{B}$:

Exercice : Conversion(s) basée(s) sur le CRT

Soit $\mathcal{B} = \{m_1, \cdots, m_k\}$ une base RNS d'intervalle dynamique $[0, M[\cap \mathbb{Z}$ et x un entier représenté dans $\mathcal{B}$ par ses résidus $(x_1, ..., x_k)$.

1. Montrer que $y = \sum_{i=1}^{k} \left| x_i \frac{m_i}{M} \right|_{m_i} \frac{M}{m_i}$ est bien défini et que $y = x + \alpha M$.
2. Donner un encadrement de α.
3. En déduire un algorithme de conversion "inexacte" de base.
4. Soit m_{sk} tel que $m_{sk} \wedge M = 1$, montrer que si $m_{sk} \geqslant k$ alors $\alpha = \left| (y - x_{sk}) M^{-1} \right|_{m_{sk}}$.
5. En déduire un algorithme de conversion "exacte" de base.

Rappels sur le RNS et FV

Conversion de base $\mathcal{B}_q = \{q_1, \ldots, q_k\} \rightarrow \mathcal{B} = \{m_1, \ldots, m_\ell\}$

- Conversion rapide mais non-exacte : x dans $\mathcal{B}_q \rightarrow |x|_q + \alpha q$ dans $\mathcal{B}$.
  $\rightsquigarrow$ Rapide mais on a un "q-overflow" $\alpha \in [0, k[\cap \mathbb{Z}$.
- Si besoin, rajouter un moduli m_{sk} à $\mathcal{B}_q$ pour corriger α efficacement (Méthode de Shenoy et Kumaresan).

Application :

On a deux bases RNS $\mathcal{B} = \{13, 17\}$, $\mathcal{B}' = \{11, 19\}$ et $m_{sk} = 5$. Convertir $x_{\mathcal{B}} = (9, 16)$, $x_{sk} = 2$ vers $\mathcal{B}'$ avec les deux méthodes vues ci dessus.

Rappels sur le RNS et FV

Conversion de base $\mathcal{B}_q = \{q_1, \ldots, q_k\} \rightarrow \mathcal{B} = \{m_1, \ldots, m_\ell\}$

- Conversion rapide mais non-exacte : x dans $\mathcal{B}_q \rightarrow |x|_q + \alpha q$ dans $\mathcal{B}$.
  $\rightsquigarrow$ Rapide mais on a un "q-overflow" $\alpha \in [0, k[\cap \mathbb{Z}$.
- Si besoin, rajouter un moduli m_{sk} à $\mathcal{B}_q$ pour corriger α efficacement (Méthode de Shenoy et Kumaresan).

Solution :

- $y = 373$
- Conversion rapide : $(10, 12)$
- $\alpha = |(y - x_{sk})221^{-1}|_5 = 1$
- Shenoy Kumaresan : $(9, 0)$

Rappels sur le RNS et FV

Contexte : $\mathcal{R}_q \cong (\mathbb{Z}/q\mathbb{Z})[X]/(X^n + 1)$

Optimisations classique *pour l'arithmétique sur...*

...les coefficients: **Residue Number Systems**

pas de contrainte sur la forme de q : $\rightarrow q = q_1 \dots q_k$ ("petits" modulis premiers entre eux et de même taille).

$$\mathbb{Z}/q\mathbb{Z} \cong \mathbb{Z}/q_1\mathbb{Z} \times \dots \times \mathbb{Z}/q_k\mathbb{Z}$$

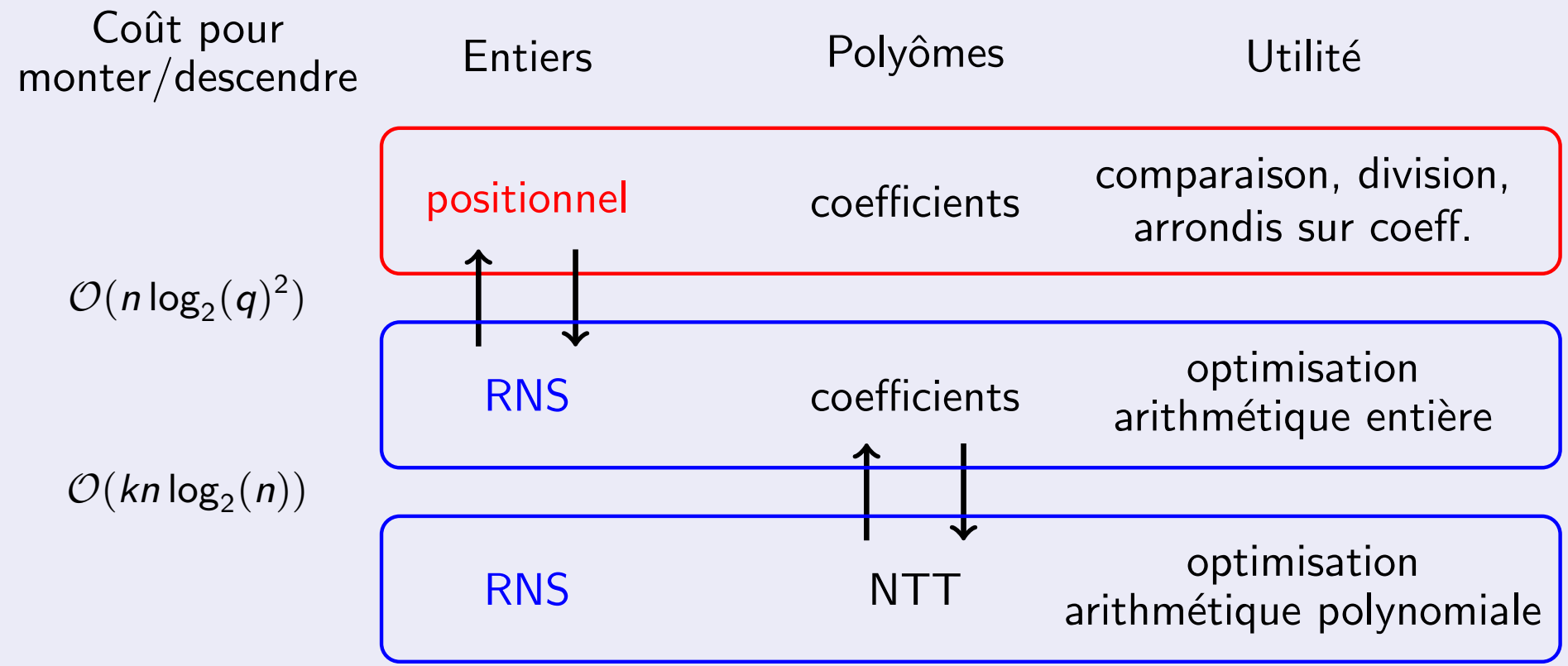
...les polynômes : **Number Theoretic Transform**

Produit polynomial optimisé avec n une puissance de 2 : $\mathcal{O}(n \log_2(n))$.

Les deux représentations sont compatibles.

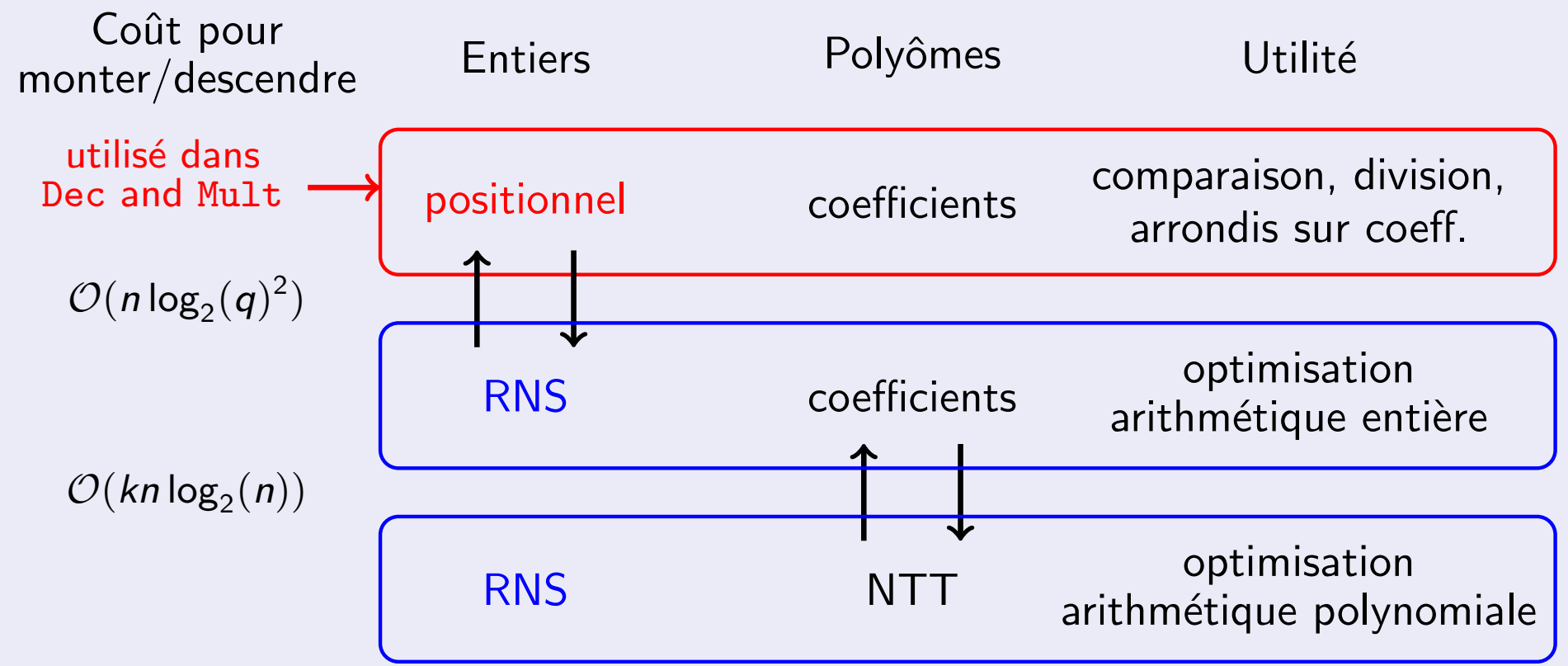
Rappels sur le RNS et FV

Tableau des représentations



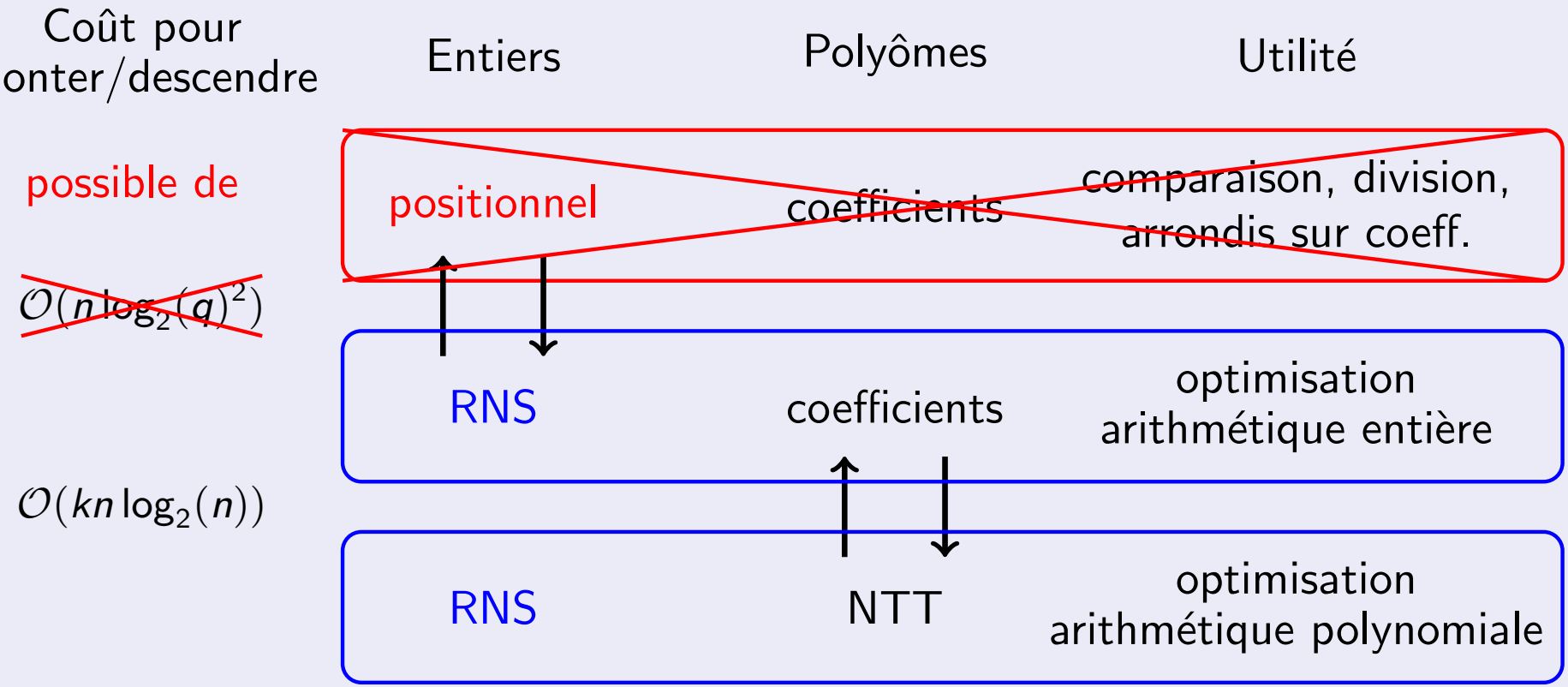
Rappels sur le RNS et FV

Tableau des représentations



Rappels sur le RNS et FV

Tableau des représentations



Rappels sur le RNS et FV

- χ_{key} et χ_{err} : des distributions "**étroites**" sur $\mathcal{R}_q$;
- $\mathcal{U}$: distribution **uniforme** sur $\mathcal{R}_q$

Géneration de clés

1. Tirer $\boldsymbol{s} \leftarrow \chi_{key}$
2. Tirer $(\boldsymbol{a}, \boldsymbol{e}) \leftarrow \mathcal{U} \times \chi_{err}$
3. Retourner $\mathbf{pk} = (\boldsymbol{p_0}, \boldsymbol{p_1}) = ([-(\boldsymbol{as} + \boldsymbol{e})]_q, \boldsymbol{a})$ (échantillon RLWE)
   $\mathbf{sk} = \boldsymbol{s}$

Chiffrement

$[\boldsymbol{m}]_t \in \mathcal{R}_t$ à chiffrer, clé publique $\mathbf{pk}$,

1. Tirer $(\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{u}) \leftarrow (\chi_{err})^2 \times \mathcal{U}$
2. Retourner $(\boldsymbol{c}_0, \boldsymbol{c}_1) = ([\Delta[\boldsymbol{m}]_t + \boldsymbol{p_0 u} + \boldsymbol{e_1}]_q, [\boldsymbol{p_1 u} + \boldsymbol{e_2}]_q)$ $(\Delta = \lfloor \frac{q}{t} \rfloor)$

$$[\boldsymbol{c}_0 + \boldsymbol{c}_1 \boldsymbol{s}]_q = \Delta[\boldsymbol{m}]_t + \boldsymbol{v} \pmod{q} \text{ avec } \boldsymbol{v}\text{: "bruit frais"}$$

Déchiffrement complètement RNS

Le déchiffrement original

$(\boldsymbol{c_0}, \boldsymbol{c_1})$ chiffrant $[\boldsymbol{m}]_t$ avec un bruit $\boldsymbol{v}$: $[\boldsymbol{c_0} + \boldsymbol{c_1 s}]_q = \Delta[\boldsymbol{m}]_t + \boldsymbol{v} + q\boldsymbol{r}$

1. normalisation : $\frac{t}{q} \cdot [\boldsymbol{c_0} + \boldsymbol{c_1 s}]_q = [\boldsymbol{m}]_t + \frac{\boldsymbol{v'}}{q} + t\boldsymbol{r}$
2. arrondis : $\lfloor \frac{t}{q} \cdot [\boldsymbol{c_0} + \boldsymbol{c_1 s}]_q \rceil = [\boldsymbol{m}]_t + \lfloor \frac{\boldsymbol{v'}}{q} \rceil + t\boldsymbol{r}$

$$\text{Si } \|\boldsymbol{v}\|_\infty = \max(|\boldsymbol{v}_i|) < B_{dec} \Rightarrow \lfloor \frac{\boldsymbol{v'}}{q} \rceil = 0$$

$$\texttt{Dec}(\boldsymbol{c}, \boldsymbol{sk}) = [\lfloor \frac{t}{q} \cdot [\boldsymbol{c_0} + \boldsymbol{c_1 s}]_q \rceil]_t = [[\boldsymbol{m}]_t + t\boldsymbol{r}]_t = [\boldsymbol{m}]_t.$$

Problème pour la représentation (**non positionelle**) RNS

Comment calculer $(\lfloor \frac{t}{q} \cdot x \rceil \bmod t)$ en RNS?

Déchiffrement complètement RNS

Calculer un arrondi en RNS

En RNS on peut faire une division exacte efficacement, on calcule donc :

$$\left\lfloor \frac{t}{q} \cdot x \right\rceil = \frac{tx - |tx|_q}{q} + b, \ (b \in \{0,1\})$$

1. **Conversion rapide** de $|tx|_q$ (en RNS) vers la base RNS $\{t\}$:

$$\texttt{FastBconv}(tx, q, \{t\}) = |tx|_q + \alpha q \bmod t \quad (\alpha \in [0, k[\cap \mathbb{Z})$$

2. $\frac{tx-(|tx|_q+\alpha q)}{q} = \lfloor \frac{t}{q} \cdot x \rceil - \alpha = \lfloor \frac{t}{q} \cdot x \rceil - E \bmod t$ (with $E = \alpha + b \leqslant k$)

Remarque : comme $tx \equiv 0 \ [t]$, on calcule seulement $\frac{-(|tx|_q+\alpha q)}{q} \bmod t$

Il y a une erreur...

Il faut corriger l'erreur E pour avoir un déchiffrement correct.

Déchiffrement complètement RNS

Corriger l'erreur

Écrivons dans $\mathbb{Z}$: $tx = \lfloor \frac{t}{q} \cdot x \rceil q + [tx]_q$

en **multipliant** par $\gamma \in \mathbb{N}$: $\left\lfloor \gamma \frac{t}{q} \cdot x \right\rceil - E = \gamma \left\lfloor \frac{t}{q} \cdot x \right\rceil + \left\lfloor \gamma \frac{[tx]_q}{q} \right\rceil - E$

Exercice

Supposons que l'on ait un **saut** $\varepsilon > 0$ i.e. $(-\frac{q}{2} + \varepsilon < [tx]_q < \frac{q}{2} - \varepsilon)$.

Montrer que si $\gamma\varepsilon \geqslant k + \frac{1}{2}$ on a: $\left[\left\lfloor \gamma \frac{t}{q} \cdot x \right\rceil - E \right]_\gamma = \left\lfloor \gamma \frac{[tx]_q}{q} \right\rceil - E$

Stratégie

1. Calculer $\lfloor \gamma \frac{t}{q} \cdot x \rceil$ modulo t **et** modulo γ
2. Utiliser le **reste centré** modulo γ pour **corriger** l'erreur.

Déchiffrement complètement RNS

$\mathtt{Dec}_{\mathtt{RNS}}((\boldsymbol{c}_0, \boldsymbol{c}_1), \boldsymbol{s}, \gamma)$

Require: $(\boldsymbol{c}_0, \boldsymbol{c}_1)$ un chiffré de $[\boldsymbol{m}]_t$, et $\boldsymbol{s}$ la clé secrète, tout deux dans la base q, un entier γ premier avec t et q.

Ensure: $[\boldsymbol{m}]_t$

1: **for** $m \in \{t, \gamma\}$ **do**
2: $\quad \boldsymbol{s}^{(m)} \leftarrow \mathtt{FastBconv}(-\gamma t(\boldsymbol{c_0} + \boldsymbol{c_1 s}), q, \{m\}) \cdot |q^{-1}|_m$
3: **end for**
4: $\tilde{\boldsymbol{s}}^{(\gamma)} \leftarrow \left[\boldsymbol{s}^{(\gamma)}\right]_\gamma$
5: $\boldsymbol{m}^{(t)} \leftarrow \left[(\boldsymbol{s}^{(t)} - \tilde{\boldsymbol{s}}^{(\gamma)}) \times |\gamma^{-1}|_t\right]_t$
6: **return** $\boldsymbol{m}^{(t)}$

Résumé

- Meilleure complexité asymptotique $\mathcal{O}(n^3) \rightarrow \mathcal{O}(n^2 \log_2(n))$.
- Flexible en terme de parallélisation.
- Borne sur le bruit modifiée : $\|\boldsymbol{v}\|_\infty < \frac{\Delta - |q|_t}{2} - \frac{k\Delta}{\gamma}$.
  (profondeur inchangée en pratique)

Multiplication homomorphique complètement RNS

Multiplication homomorphique originale de (c_0, c_1) par (c'_0, c'_1)

1. Produit : $(\tilde{c}_0, \tilde{c}_1, \tilde{c}_2) = (c_0 c'_0, c_0 c'_1 + c'_0 c_1, c_1 c'_1)$ dans $\mathbb{Z}$ (reconstruire)
2. Division + Arrondis : $\hat{c}_i = \lfloor \frac{t}{q} \cdot \tilde{c}_i \rceil$
   $$\rightsquigarrow [\hat{c}_0 + \hat{c}_1 s + \hat{c}_2 s^2]_q = \Delta [m_1 m_2]_t + v'' \bmod q$$
3. Relinéarisation : $(\hat{c}_0 + \hat{c}_2 s^2, \hat{c}_1) \xrightarrow{s \text{ private}} (\hat{c}_0 + \hat{c}_2(s^2 + e + as), \hat{c}_1 - a\hat{c}_2)$
   - Bruit trop "gros" $(\|\hat{c}_2 \times e\|_\infty < q/2 \times nB_{\mathtt{err}})$ → Solution originale :
   - Décomposer $\hat{c}_2 = b_0 + b_1\omega + \ldots + b_\ell \omega^{\ell-1}$ en base ω
   - Clé publique de relinéarisation: $([s^2 \cdot (1, \omega, \ldots, \omega^{\ell-1}) + \vec{e} + \vec{a}s]_q, -\vec{a})$
   - Bruit plus faible $\|(b_0, b_1, \ldots, b_\ell) \cdot \vec{e}\|_\infty < \ell\omega/2 \times nB_{\mathtt{err}}$

Problèmes pour la représentation RNS

Reconstruction dans $\mathbb{Z}$, division et arrondis, décomposition en base ω...

Multiplication homomorphique complètement RNS

- Problème 1 : calculer le produit $(\tilde{\boldsymbol{c}}_0, \tilde{\boldsymbol{c}}_1, \tilde{\boldsymbol{c}}_2)$ in $\mathbb{Z}$.
  $\longrightarrow$ Solution : $\|\tilde{\boldsymbol{c}}_i\|_\infty <\sim nq^2$: ne pas reconstruire dans $\mathbb{Z}$,
  1. Conversion rapide des résidus dans la deuxième base $\mathcal{B}_{sk} = \mathcal{B} \cup \{m_{sk}\}$;
  2. Calculer $(\tilde{\boldsymbol{c}}_0, \tilde{\boldsymbol{c}}_1, \tilde{\boldsymbol{c}}_2)$ in $q \cup \mathcal{B}_{sk}$.

Multiplication homomorphique complètement RNS

Problème 1: calculer les produits $(\tilde{c}_0, \tilde{c}_1, \tilde{c}_2)$

$\|\tilde{c}_i\|_\infty <\sim nq^2$: pas de reconstruction dans $\mathbb{Z}$.
$\rightarrow$ Juste prendre une base plus grande que $\{q_1, \ldots, q_k\}$

Solution : introduire une deuxième base $\mathcal{B}$

Étape	base $q_1, \ldots, q_k$		base $\mathcal{B} \cup \{\tilde{m}\}$
0	c_i, c'_j		
1	c_i, c'_j	$\xrightarrow[\text{non-exacte}]{\text{conversion rapide}}$	$c_i + qu_i, c'_j + qu'_j$ (à réduire)
2			$\hat{c}_i \leftarrow \texttt{MRed\_q}_{\tilde{m}}(c_i + qu_i)$
3	$\tilde{c}_0 \leftarrow c_0 \times c'_0$, etc		$\tilde{c}_0 \leftarrow \hat{c}_0 \times \hat{c}'_0$, etc

- Seulement des conversions de base RNS **non-exactes mais rapides**
- **cheap** Réduction de Montgomery des q-overflows $u'_i s$

Multiplication homomorphique complètement RNS

- Problème 1 : calculer le produit $(\tilde{\boldsymbol{c}}_0, \tilde{\boldsymbol{c}}_1, \tilde{\boldsymbol{c}}_2)$ in $\mathbb{Z}$.
  $\longrightarrow$ Solution : $\|\tilde{\boldsymbol{c}}_i\|_\infty <\sim nq^2$: ne pas reconstruire dans $\mathbb{Z}$,
  1. Conversion rapide des résidus dans la deuxième base $\mathcal{B}_{sk} = \mathcal{B} \cup \{m_{sk}\}$;
  2. Calculer $(\tilde{\boldsymbol{c}}_0, \tilde{\boldsymbol{c}}_1, \tilde{\boldsymbol{c}}_2)$ in $q \cup \mathcal{B}_{sk}$.

- Problème 2: division et arrondis des $\hat{\boldsymbol{c}}_i = \lfloor \frac{t}{q} \cdot \tilde{\boldsymbol{c}}_i \rceil$ en RNS.
  $\longrightarrow$ Solution: partie entière au lieu de l'arrondis dans $\mathcal{B}_{sk}$,
  1. $\texttt{FastBconv}(\tilde{\boldsymbol{c}}_i, q, \mathcal{B}_{sk})$ et calcule de la partie entière dans $\mathcal{B}_{sk}$;
  2. $\hat{\boldsymbol{c}}_i \leftarrow \texttt{FastBconvSK}(\tilde{\boldsymbol{c}}_i, \mathcal{B}_{sk}, q)$ (pas de multiple du module de $\mathcal{B}$).

Multiplication homomorphique complètement RNS

Problème 2 : division et arrondis $\hat{c}_i = \lfloor \frac{t}{q} \cdot \tilde{c}_i \rceil$ en RNS

Contexte $\neq$ déchiffrement, aucune garantie d'avoir un saut ϵ.
$\Rightarrow$ On ne **peut pas** obtenir un **arrondi RNS correct** avec la γ-correction.

Solution : partie entière au lieu de l'arrondi

Étape	base $q_1, \ldots, q_k$		base $\mathcal{B} \cup \{m_{sk}\}$
0	$\tilde{c}_i$		$\tilde{c}_i$
1	$t\tilde{c}_i$	$\xrightarrow[\text{non-exacte}]{\text{conversion rapide}}$	$\lvert t\tilde{c}_i \rvert_q + q\tilde{u}_i$
2			$\hat{c}_i \leftarrow \frac{t\tilde{c}_i - (\lvert t\tilde{c}_i \rvert_q + q\tilde{u}_i)}{q}$ in $\mathcal{B} \cup \{m_{sk}\}$
3	$\hat{c}_i = \lfloor \frac{t}{q} \cdot \tilde{c}_i \rceil + e_i$	$\xleftarrow[\text{exacte}]{\text{conversion rapide}}$	$\hat{c}_i, \lvert \hat{c}_i \rvert_{m_{sk}}$
	$(\lVert e_i \rVert_\infty \leqslant k)$		(on utilise $\lvert \hat{c}_i \rvert_{m_{sk}}$ pour corriger l'overflow)

Partie entière $\Rightarrow$ le **bruit** augmente plus vite.

Multiplication homomorphique complètement RNS

- Problème 1 : calculer le produit $(\tilde{\boldsymbol{c}}_0, \tilde{\boldsymbol{c}}_1, \tilde{\boldsymbol{c}}_2)$ in $\mathbb{Z}$.
  $\rightarrow$ Solution : $\|\tilde{\boldsymbol{c}}_i\|_\infty <\sim nq^2$: ne pas reconstruire dans $\mathbb{Z}$,
  1. Conversion rapide des résidus dans la deuxième base $\mathcal{B}_{sk} = \mathcal{B} \cup \{m_{sk}\}$;
  2. Calculer $(\tilde{\boldsymbol{c}}_0, \tilde{\boldsymbol{c}}_1, \tilde{\boldsymbol{c}}_2)$ in $q \cup \mathcal{B}_{sk}$.

- Problème 2: division et arrondis des $\hat{\boldsymbol{c}}_i = \lfloor \frac{t}{q} \cdot \tilde{\boldsymbol{c}}_i \rceil$ en RNS.
  $\rightarrow$ Solution: partie entière au lieu de l'arrondis dans $\mathcal{B}_{sk}$,
  1. $\texttt{FastBconv}(\tilde{\boldsymbol{c}}_i, q, \mathcal{B}_{sk})$ et calcule de la partie entière dans $\mathcal{B}_{sk}$;
  2. $\hat{\boldsymbol{c}}_i \leftarrow \texttt{FastBconvSK}(\tilde{\boldsymbol{c}}_i, \mathcal{B}_{sk}, q)$ (pas de multiple du module de $\mathcal{B}$).

- Problem 3: décomposer $\hat{\boldsymbol{c}}_2$ en base ω dans la base q.
  $\rightarrow$Solution: décomposer $\hat{\boldsymbol{c}}_2$ "façon RNS",
  1. $\hat{\boldsymbol{c}}_2 = \left(|\hat{\boldsymbol{c}}_2 \cdot \frac{q_1}{q}|_{q_1}, \cdots, |\hat{\boldsymbol{c}}_2 \cdot \frac{q_k}{q}|_{q_k} \right)$;
  2. $\mathbf{evk}_{RNS} = \left(\left[\left(|\boldsymbol{s}^2 \frac{q}{q_1}|_q, \cdots, |\boldsymbol{s}^2 \frac{q}{q_k}|_q \right) + \vec{\boldsymbol{e}} + \vec{\boldsymbol{a}}\,\boldsymbol{s} \right]_q, \vec{\boldsymbol{a}} \right)$.

Multiplication homomorphique complètement RNS

Problème 3 : Accès à un système positionel

Décomposer $\hat{c}_2 = b_0 + b_1\omega + \ldots + b_\ell\omega^{\ell-1}$ en base ω (positional).

Rappel : remplacer $\hat{c}_2 \times e \overset{\|.\|}{\sim} nqB_{\mathtt{err}} \rightsquigarrow (b_0, b_1, \ldots, b_\ell) \cdot \vec{e} \overset{\|.\|}{\sim} n\ell\omega B_{\mathtt{err}}$

Solution : utiliser la representation RNS...

$$\begin{array}{rcll} \hat{c}_2 & = & b_0 + b_1\omega + \ldots + b_\ell\omega^{\ell-1} & (b_i = \|\lfloor \hat{c}_2\omega^{-i} \rceil\|_\omega) \\ \hat{c}_2 & = & d_1\frac{q}{q_1} + d_2\frac{q}{q_2} + \ldots + d_k\frac{q}{q_k} \mod q & (d_i = |\hat{c}_2\frac{q_i}{q}|_{q_i}) \end{array}$$

$$\|b_i\|_\infty \sim \|d_i\|_\infty \Rightarrow \|(b_0, b_1, \ldots, b_\ell) \cdot \vec{e}\|_\infty \sim \|(d_1, d_2, \ldots, d_k) \cdot \vec{e}\|_\infty$$

$\Rightarrow$ la borne $\ell\omega \times nB_{\mathtt{err}}$ devient $k\omega \times nB_{\mathtt{err}}$

$\rightarrow$ si $\omega \sim q_i$ (i.e. $\ell = k$), la representation RNS a un effet équivalent.

Eval. key: $s^2 \cdot (1, \omega, \ldots, \omega^{\ell-1}) + \vec{e} + \vec{a}\,s \rightsquigarrow s^2 \cdot (\frac{q}{q_1}, \ldots, \frac{q}{q_k}) + \vec{e} + \vec{a}\,s$

Full RNS variant of FV multiplication

Résumé

- Opérations coûteuses dans $\mathbb{Z} \rightsquigarrow$ Conversions de base rapide.
- Le bruit croit de manière équivalente (pas d'impact sur la profondeur).
- Même nombre de produit polynomiaux $\Rightarrow$ Même complexité asymptotique.
- Meilleure complexité pour les opérations sur les coefficients.
- Peut être complètement parallélisé.

Implentation Logicielle

- C++ utilisant NFLlib (pour l'arithmétique polynomiale dans $\mathcal{R}$ avec NTT),
- comparée avec [a] une approche standart avec NFLlib+GMP 6.1.0,
- Ordinateur portable avec Fedora 22, un i7-4810MQ CPU @ 2.80GHz, g++ v5.3.1, Hyper-Threading et Turbo Boost désactivés.

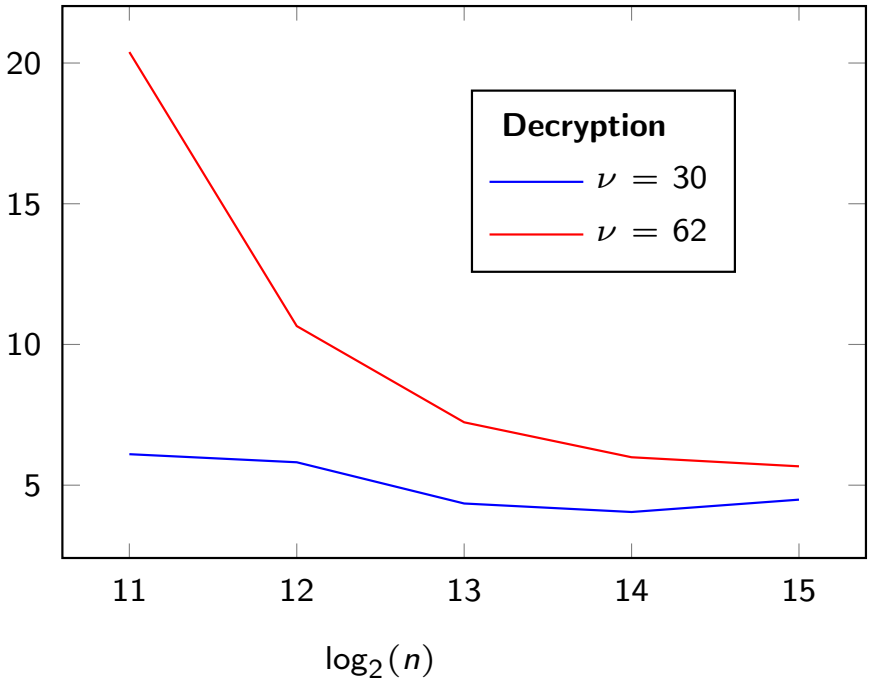
[a] `https://github.com/CryptoExperts/FV-NFLlib`

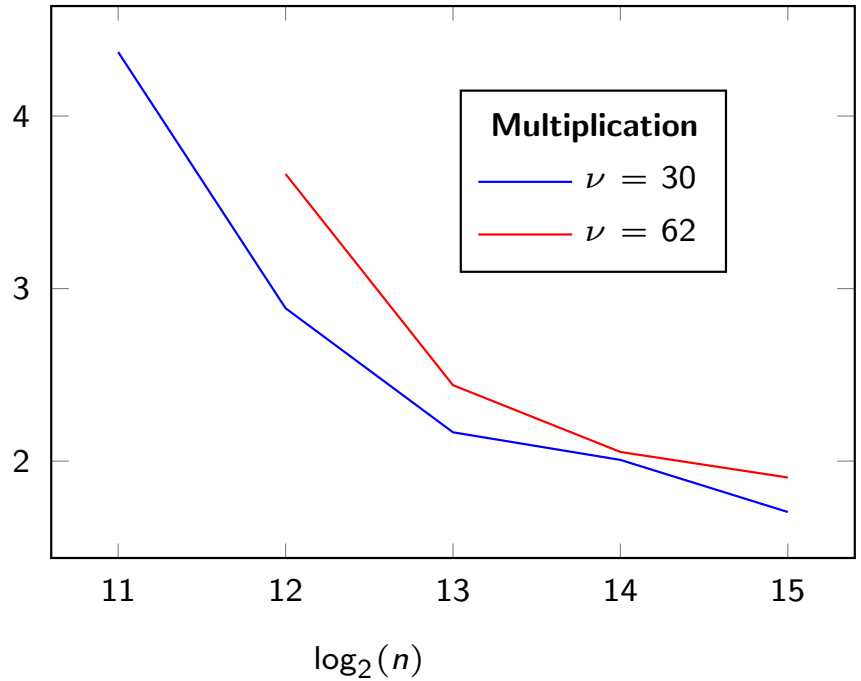
Résultats - Facteurs d'accélération

ν taille des modulis (bit)	$\log_2(n)$	11	12	13	14	15
30	k	3	6	13	26	53
62	k	1	3	6	12	25

$t = 2^{10}$

$\gamma = 2^8$ (suffisant en pratique)





$n \nearrow \Rightarrow$ les NTTs dominent le coût de calcul $\Rightarrow$ accélération $\searrow$.

Conclusion

- La conversion vers des algorithmes n'utilisant que du RNS permet d'obtenir une amélioration significative des temps de calcul dans FV.
- Le bruit croit plus rapidement en théorie mais il n'y a pas de différence significative en pratique.
- Permet d'envisager des implémentations adaptées sur des périphériques exploitant pleinement le potentiel de parallélisation de ces algorithmes, tels que des GPUs et des FPGAs.