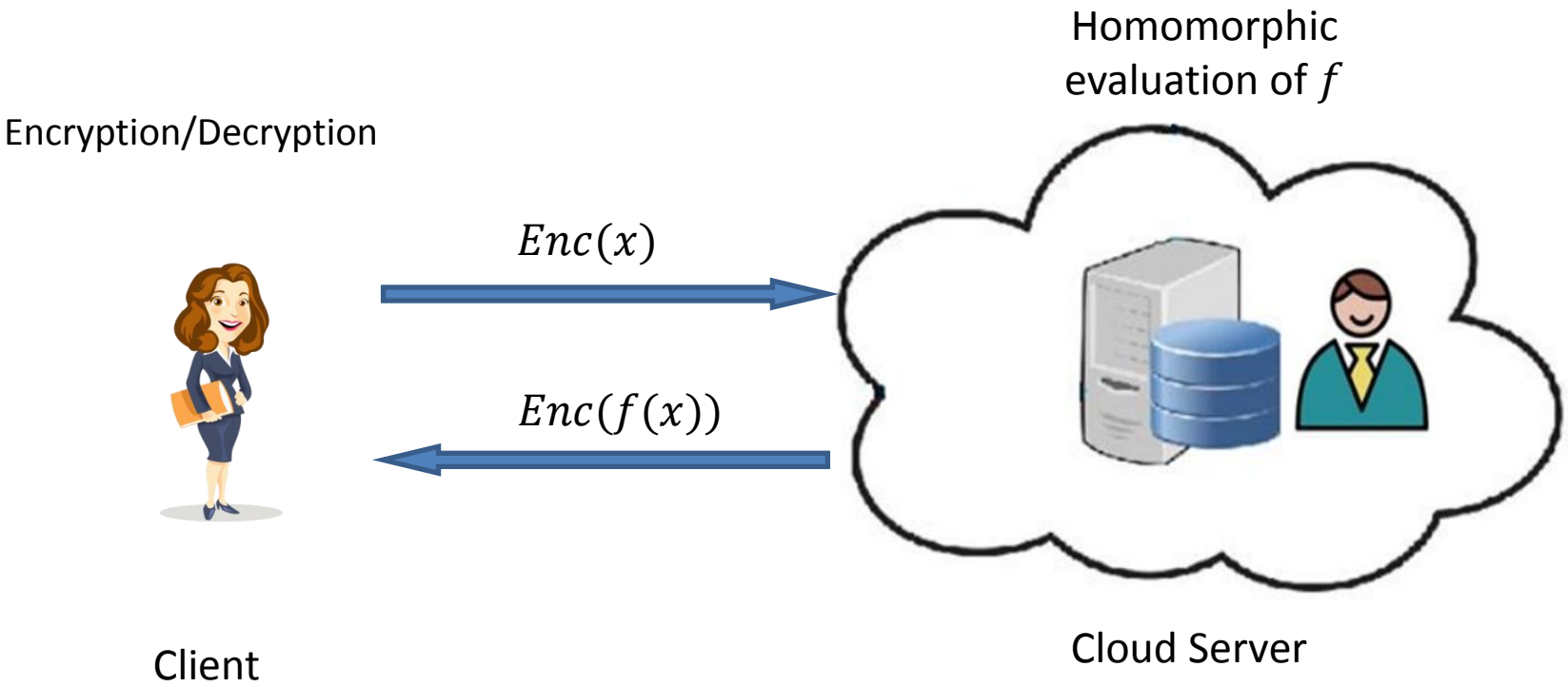
High-Performance FV Somewhat Homomorphic Encryption on GPUs: An Implementation using CUDA

Ahmad Al Badawi
ahmad@u.nus.edu
National University of Singapore (NUS)

Sept 10th 2018 – CHES 2018

FHE – The holy grail of Cryptography

- FHE enables computing on encrypted data without decryption [GB2009]
- Challenge: requires enormous computation



How the problem is being tackled?

- Algorithmic methods:
  - New FHE schemes
  - Plaintext packing (1D, 2D, …)
  - Encoding schemes
  - Approximated computing
  - Squashing the target function
  - DAG optimizations for the target circuit
- Acceleration methods:
  - Speedup FHE basic primitives (KeyGen, Enc, Dec, Add, Mul)
  - Modular Algorithms
  - Parallel Implementations
  - Hardware implementations: GPUs, FPGAs and probably ASICs

Our Contributions

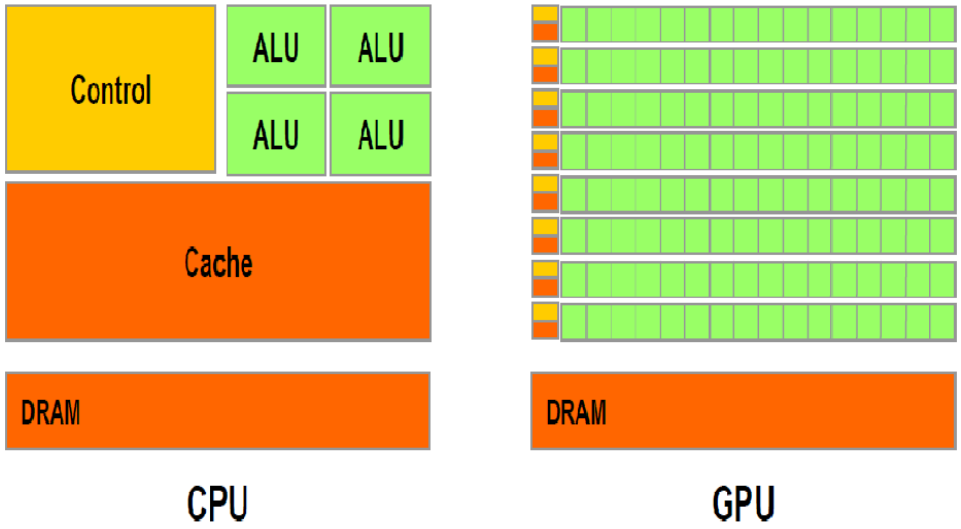
1. Implementation of FV RNS on GPUs

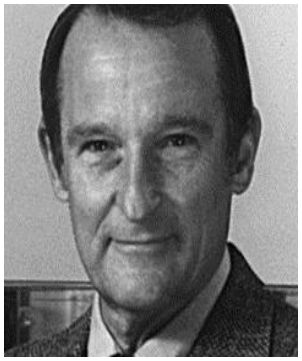
2. Introducing a set of CUDA optimizations

3. Benchmarking with state-of-the-art implementations

Why GPUs for FHE?

- GPU +
  - Naturally available
  - many computing cores
  - Developer friendly (FPGA, ASIC)
- FHE +
  - Huge level of parallelism





Seymour Cray
1925-1996

"If you were plowing a field, which would you rather use? Two strong oxen or 1024 chickens?"

Textbook FV

- Basic mathematical structure is $R: \mathbb{Z}[x]/(x^n + 1)$
  - Plaintext space: $R_t: \mathbb{Z}_t[x]/(x^n + 1)$
  - Ciphertext space: $R_q: \mathbb{Z}_q[x]/(x^n + 1)$
- Public key: $(pk_0, pk_1) \in R_q$
- Secret key: $(sk) \in R_q$
- $c = Enc(m): \left(\left[\left\lfloor \frac{q}{t} \right\rfloor m + pk_0 u + e_0\right]_q, [pk_1 u + e_1]_q\right)$
- $m = Dec\ (c): \left[\left\lfloor \frac{t}{q} \right\rfloor [c_0 + c_1 sk]_q\right]_t$
- $c^+ = Add(c_0, c_1): ([c_{00} + c_{10}]_q, [c_{01} + c_{11}]_q)$

Textbook FV (cont.)

- $c^{\times} = Mul(c_0, c_1, evk)$:
  1. Tensor product:

$$v_0 = \left[\lfloor \frac{t}{q}\, c_{00}c_{10} \rceil\right]_q \quad , \quad v_2 = \left[\lfloor \frac{t}{q}\, c_{01}c_{11} \rceil\right]_q$$

$$v_1 = \left[\lfloor \frac{t}{q}(c_{00}c_{11} + c_{01}c_{10}) \rceil\right]_q$$

  2. Base decomposition:

$$v_2 = \sum_{i=0}^{l} v_2^{(i)} w^i \ , l =$$

  2. Relinearization:

$$c^{\times} = \left[v_j + \sum_{i=0}^{l} evk_{ij} \cdot v_2^{(i)}\right]_q \ , j \in \{0,1\}$$

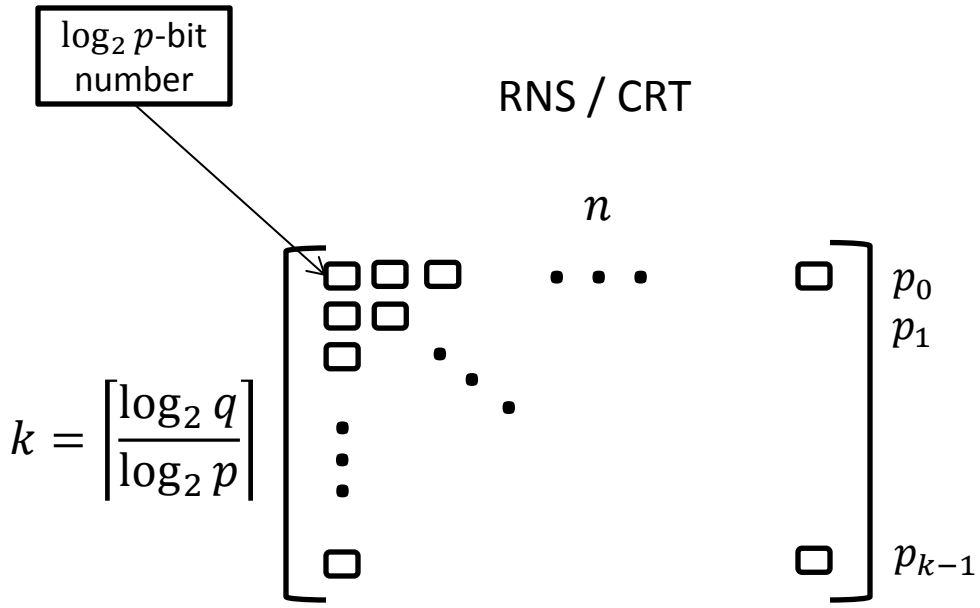
Implementation Requirements

- Polynomial arithmetic in cyclotomic rings
- Large polynomial degree (a few thousands)
  - Power-of-2 cyclotomic
  - Addition/Subtraction: $\mathcal{O}(n)$
  - Multiplication: $\mathcal{O}(\mathrm{n}\log n)$
- Large coefficients $\in \mathbb{Z}_q$ (a few hundreds of bits)
  - Modular algorithms (RNS)
- Extra non-trivial operations:
  - Scaling-and-round $\lfloor \frac{t}{q} x \rceil$
  - Base decomposition

Polynomial Arithmetic

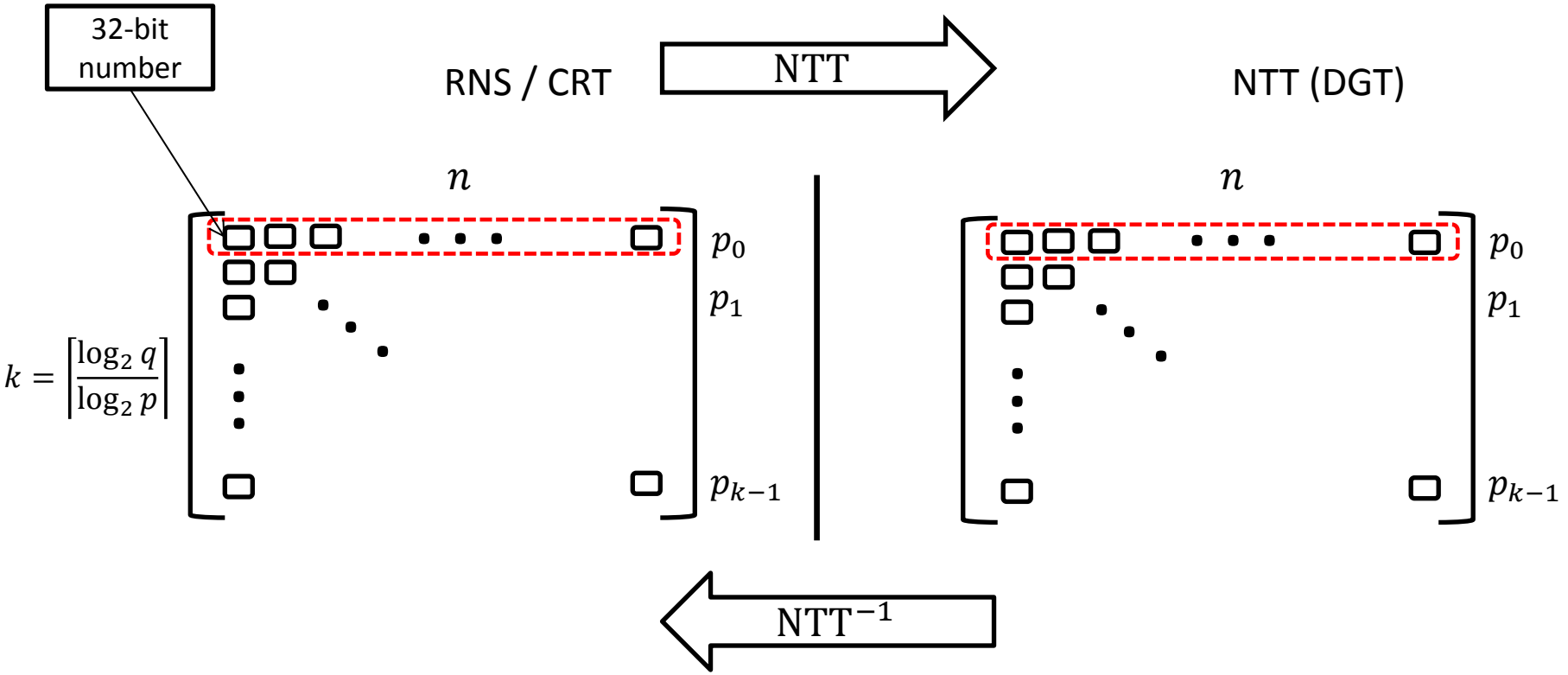
CRT:

$q = \prod_{i=0}^{k-1} p_i$, where p_i is a prime



Addition/Subtraction: component-wise add/sub modulo p_i

Polynomial Arithmetic (cont.)



Addition/Subtraction/Multiplication: component-wise add/sub/mul modulo p_i

DFT, NTT, DWT, DGT…?

	Pros	Cons
DFT	- Well-established - Several efficient libraries to use	- Floating point errors increase as (n & p_i's) increase - Reduce precision (smaller p_i's) => longer RNS matrix => more DFTs
NTT	- Exact	- Transform length ($2n$)
DWT	- Exact	- Only power-of-2 cyclotomics - Transform length (n)
DGT	- Exact - Transform length ($\frac{n}{2}$) - 50% Less interaction with memory	- Only power-of-2 cyclotomics - Gaussian Arithmetic (larger number of multiplications ~(30% - 40%)

- We use DGT in our implementation

Efficient DGT/NTT/DWT on GPU?

- Better to store w^{ji} in lookup table.
  - LUT can be stored in GPU texture memory (which is limited on GPU)
  - DWT LUT are $\mathcal{O}(n)$
  - DGT LUT are $\mathcal{O}(\frac{n}{2})$
- Compute in $GF(\hat{p})$ or in $GF(p_i)$?
  - We found it is better to do it $GF(p_i)$.
  - Why? (see next)

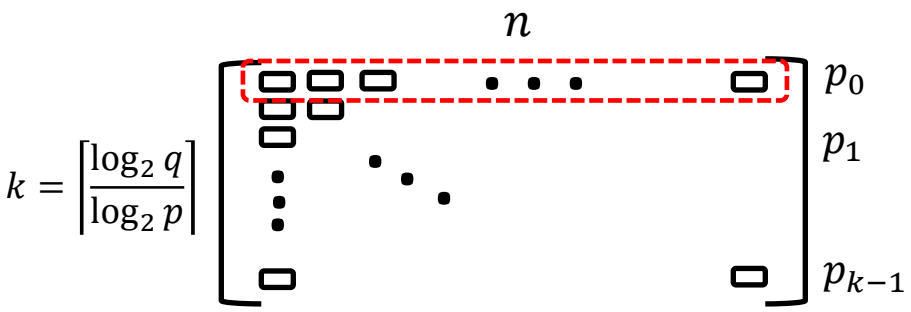
$A = \mathrm{NTT}(a)$ s.t.

$$A_j = \sum_{i=0}^{n-1} a_i w^{ji} \mod q$$

$a = \mathrm{NTT}^{-1}(A)$ s.t.

$$a_i = n^{-1} \sum_{j=0}^{n-1} A_j w^{-ij} \mod q$$

Compute in $GF(\hat{p})$ or in $GF(p_i)$?



$GF(\hat{p})$

$\hat{p}$: 64-bit prime (should fit in one word)

$p \leq \left\lfloor \sqrt{\frac{\hat{p}}{2n}} \right\rfloor$ (one multiplication)

n	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
$\log_2 p$	26	25	25	24	24

- Longer RNS matrix => more NTTs
- Size double (32-bit => 64-bit)
- Supports limited number of operations in NTT domain

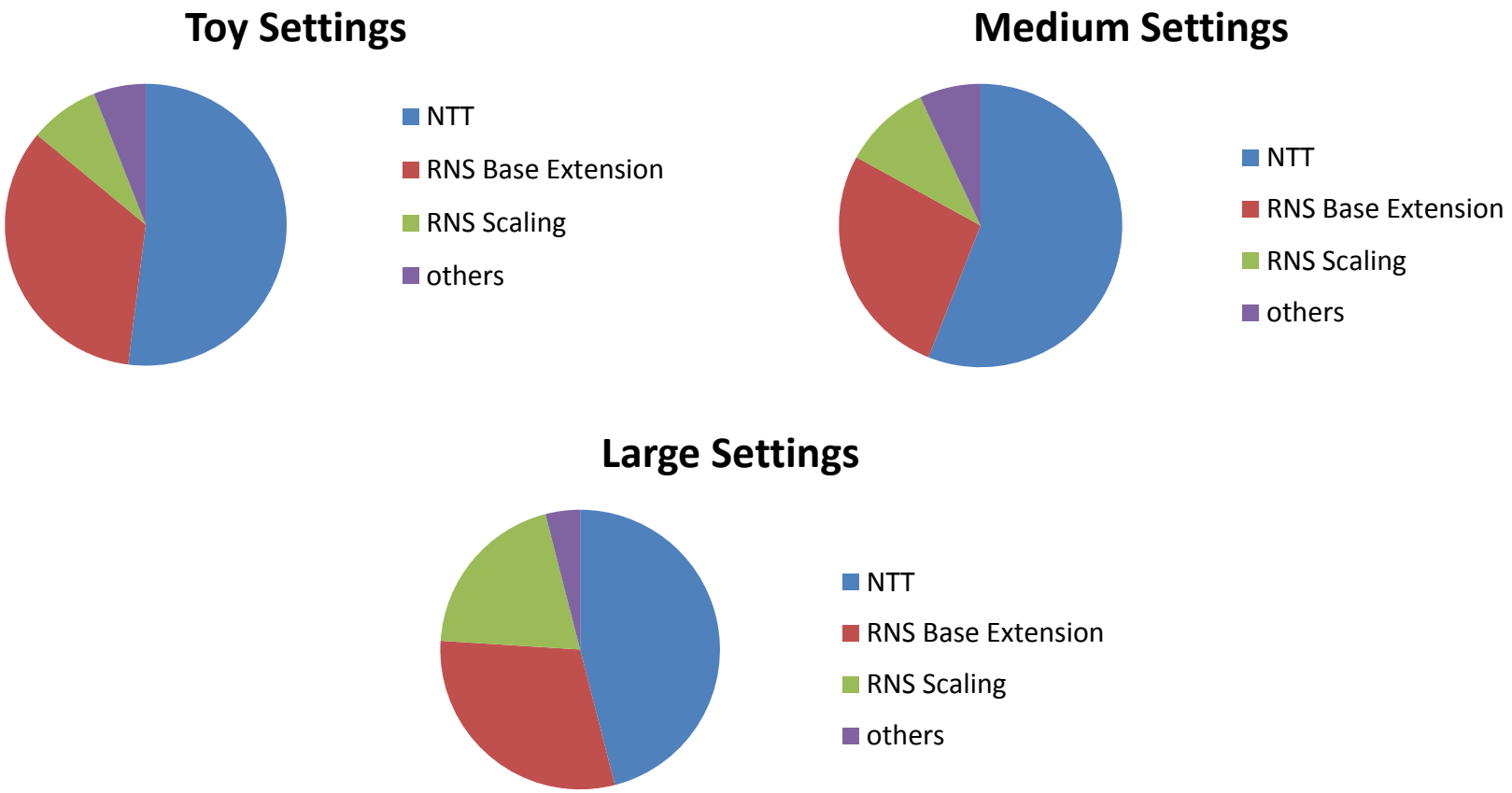
$GF(p_i)$

p: word-size prime (can be 64-bit)

- Shorter RNS matrix => Less NTTs
- No size doubling
- Supports unlimited number of operations in NTT domain

But, is NTT/DWT/DGT performance-critical?

Breakdown of homomorphic multiplication (AND) in the BFV FHE scheme



Halevi, Shai, Yuriy Polyakov, and Victor Shoup. "An Improved RNS Variant of the BFV Homomorphic Encryption Scheme." (2018).

Computing CRT on GPU?

- At least two methods:
  - Classic algorithm
  - Garner's algorithm

- $\mathrm{CRT}(a, \{p_i\})$:

$$(a_0, \dots, a_{k-1}) = a \bmod p_i$$

- $\mathrm{CRT}^{-1}(a_0, \dots, a_{k-1}) = a$ s.t.

$$a = \sum_{i=0}^{k-1} \frac{q}{p_i} \left(\left(\frac{q}{p_i} \right)^{-1} a_i \pmod{p_i} \right) \pmod{q}$$

where $q = \prod_{i=0}^{k-1} p_i$

	Classic	**Garners**
LUT	k^2	$\frac{k(k-1)}{2}$
Thread Divergence	Non tractable	Nil

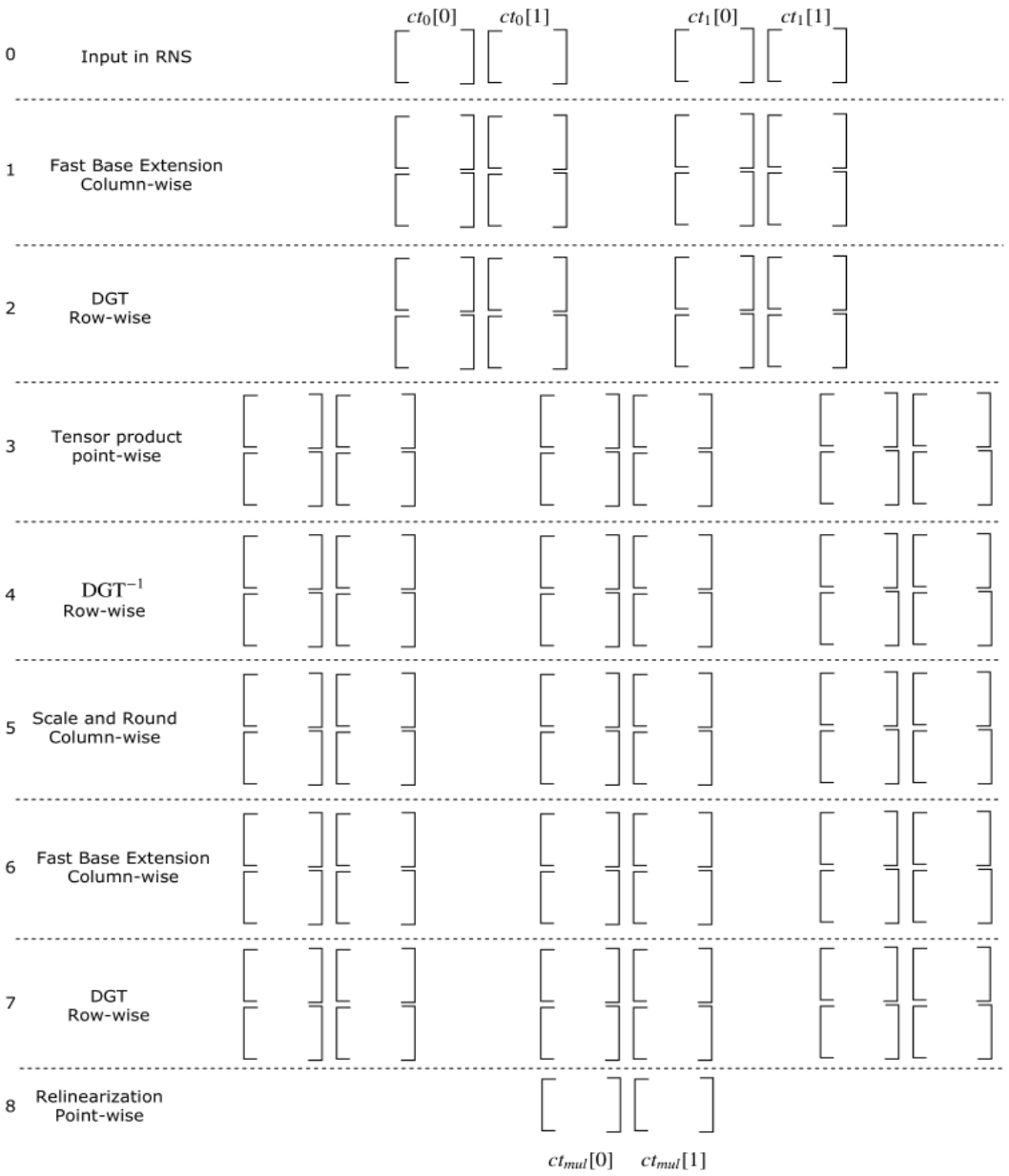
- Is CRT critical to performance?
  - No!

RNS tools

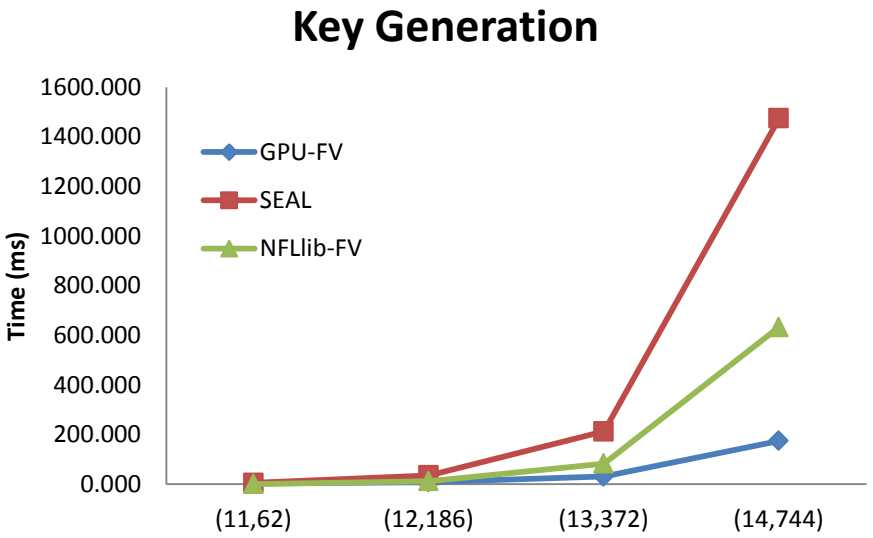
- Useful to:
  - Remain in RNS representation
  - No costly multi-precision arithmetic
- Two basic operations:
  - Scale-and-round
  - Base decomposition
- Adopted from (BEHZ2016*) scheme
- Are RNS tools critical to performance?
  - Extremely critical

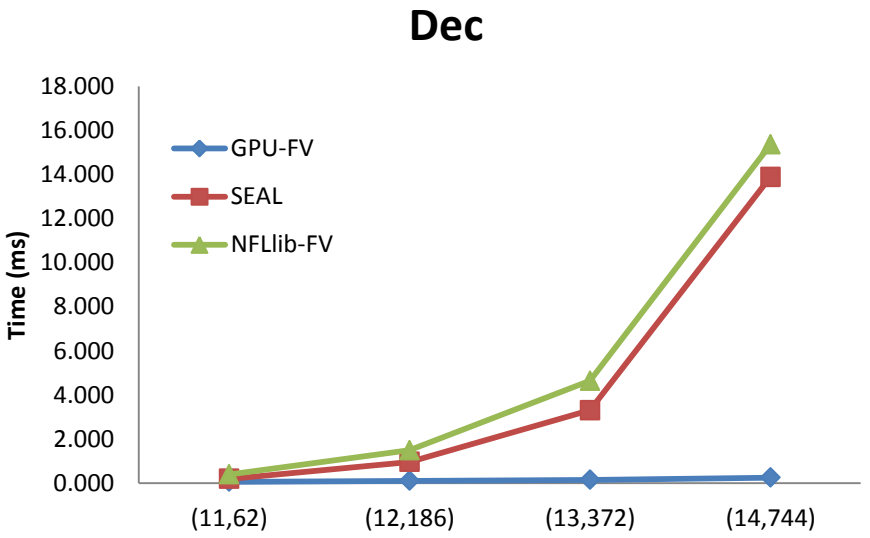
* Bajard, Jean-Claude, et al. "A full RNS variant of FV like somewhat homomorphic encryption schemes." *International Conference on Selected Areas in Cryptography*. Springer, Cham, 2016.

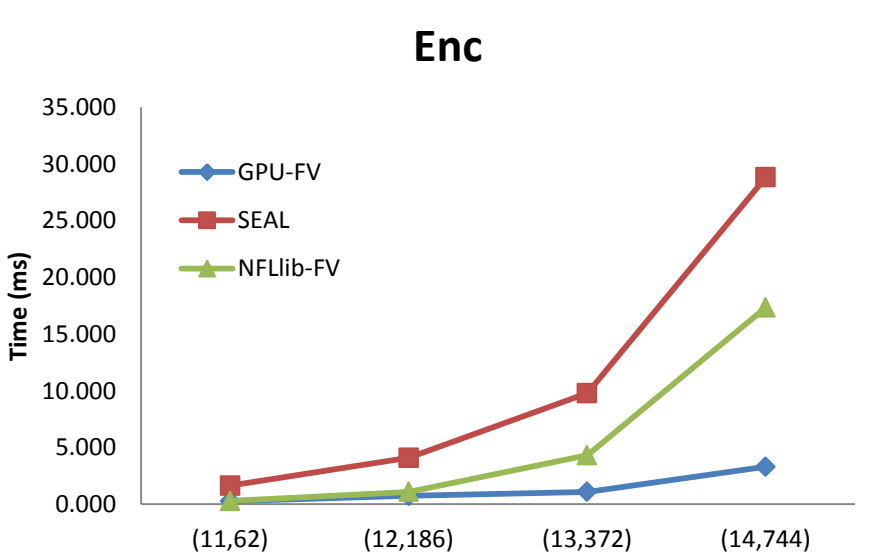
FV_RNS Homomorphic Multiplication

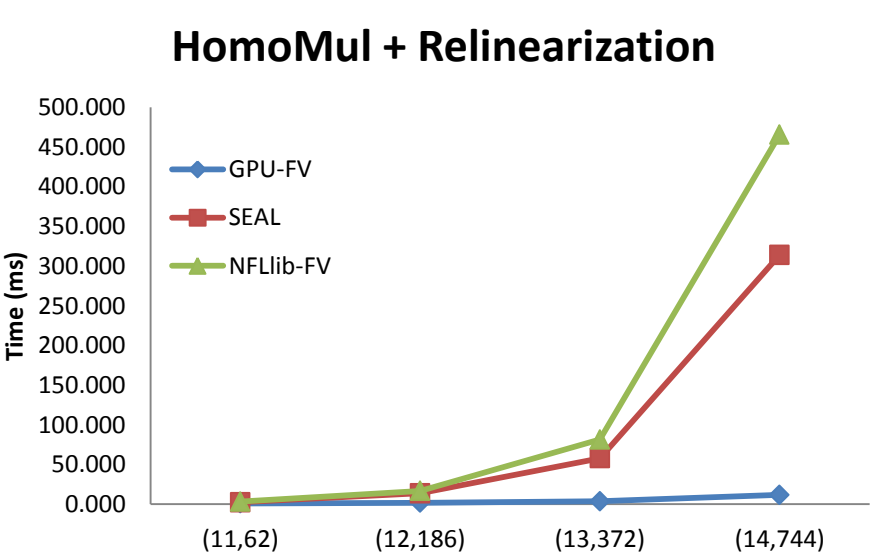


Benchmarking Results









Which FV RNS variant to Implement?

- Two RNS variants of FV
  - BEHZ
  - HPS
- Answer can be found in:
  - Al Badawi, Ahmad, et al. "Implementation and Performance Evaluation of RNS Variants of the BFV Homomorphic Encryption Scheme." *IACR Cryptology ePrint Archive* 2018 (2018): 589.

Thank You

Questions?

Ahmad Al Badawi
ahmad@u.nus.edu