# Accelerating NTRU based Homomorphic Encryption using GPUs

Wei Dai, Yarkın Doröz, Berk Sunar

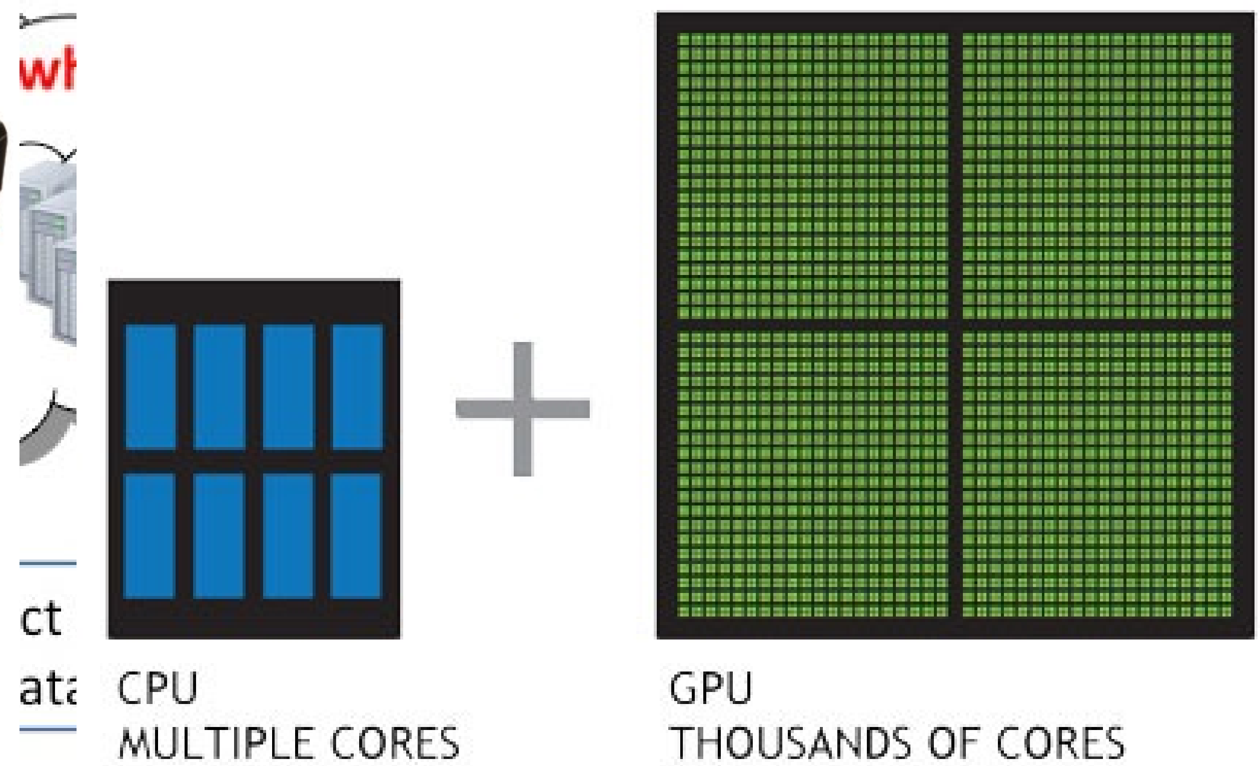
Worcester Polytechnic Institute
100 Institute Road, Worcester, MA, USA, 01609

# Outline:

- Motivations
- NTRU
- GPU NTRU
- Evaluation Results

# Motivation

- Homomorphic Encryption is not efficient
- Speedup computations with GPUs

# NTRU

- Keys, cipher texts are polynomials
  - n : polynomial degree
  - q : prime modulus
  - l : length of q
- Operations are performed in $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$
- Previously, implemented with NTL Library in C++

# GPU NTRU

- Core problem: *polynomial multiplications*
- How to create parallelism?
  - Chinese Remainder Theorem (CRT)
  - Number Theory Transform (NTT)
- Fast algorithms
- Memory access

# GPU NTRU: Multiplication

- AES: 32768 degree, 1271-bit coefficients
- PRINCE: 16384 degree, 575-bit coefficients
- The Strassen's NTT based integer multiplication algorithm

**Algorithm 1** Polynomial Multiplication

**Input:** Polynomials $a$, $b$ with $(n, \log(q))$
**Output:** Polynomial $c$ with $(2n, \log(nq^2))$

$\cancel{\{a_i\} = \mathsf{CRT}(a), \{b_i\} = \mathsf{CRT}(b)}$
$\{A_i\} = \mathsf{NTT}(\{a_i\}), \{B_i\} = \mathsf{NTT}(\{b_i\})$
$\{C_i\} = \{A_i\} \cdot \{B_i\}$
$\{c_i\} = \mathsf{INTT}(\{C_i\})$
$\cancel{c = \mathsf{ICRT}(\{c_i\})}$

# GPU NTRU: CRT

$$\texttt{CRT}: x \longrightarrow \{x \bmod p_0, x \bmod p_1, \cdots, x \bmod p_{l-1}\}$$

- Reduce coefficient size

  1271-bit $\longrightarrow$ 42-bit      575-bit $\longrightarrow$ 25-bit

- Size and number of $p_i$ are decided automatically
  - in different circuit levels
  - only according to n and q
  - as level increases, computation goes faster
- Rules will be explained later

# GPU NTRU: ICRT

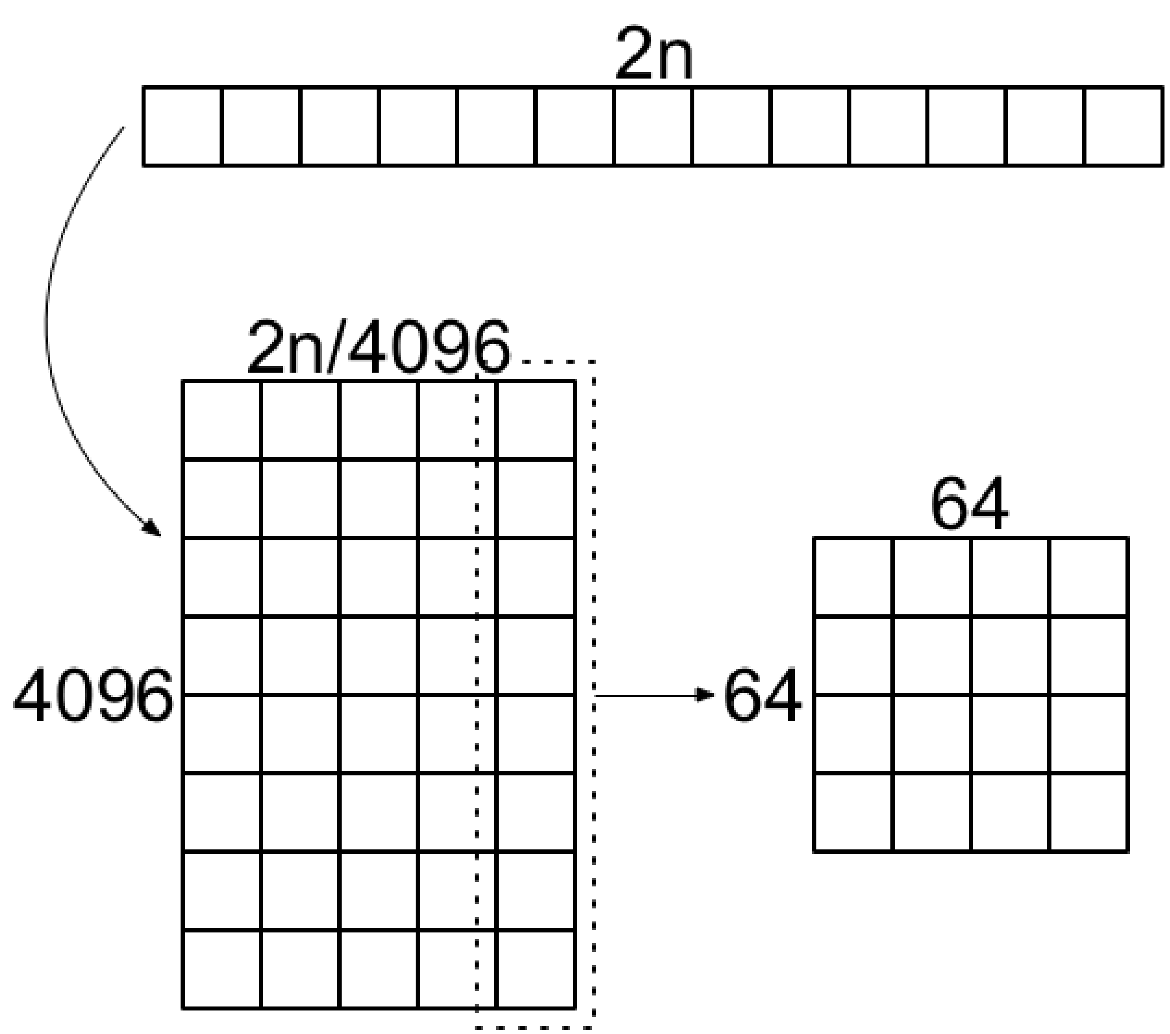
$$\mathtt{ICRT}: x = \sum_{i=0}^{l-1} \left(\frac{M}{p_i}\right) \cdot \left( \left(\frac{M}{p_i}\right)^{-1} \cdot x_i \mod p_i \right) \mod M$$

$$M = \prod_{i=0}^{l} p_i$$

- Modified ICRT scheme
  - avoid large integer multiplication
  - avoid large integer modular reduction
- NVIDIA GPU Constant memory

# GPU NTRU: NTT

- Emmart and Weems' approach
- 2n-point coefficient-wise NTT (padding with 0)
- Four-step Cooley-Tukey NTT iterations:



# GPU NTRU: NTT

- Over the finite field $\mathbb{Z}/P\mathbb{Z}$, $P = \texttt{0xffffffff00000001}$
- Prime numbers:

$$P > n \cdot p_i^2 \qquad \prod_{i=0}^{l-1} p_i > n \cdot q^2$$

- Memory arrangement:
  - coalesced global memory access
  - shared memory as buffers
  - registers for arithmetic operations

# GPU NTRU: Relinearization

- Input: cipher text $c(x)$, evaluation keys $\{EK_i(x)\}$
- Take the *i*-th bit of coefficients in $c(x)$
- Binary polynomials $\tilde{c}_i(x)$
- Output:

$$\tilde{c}(x) = \sum_{i=0}^{l-1} \tilde{c}_i(x) \cdot EK_i(x)$$

- Thousands of polynomial multiplications

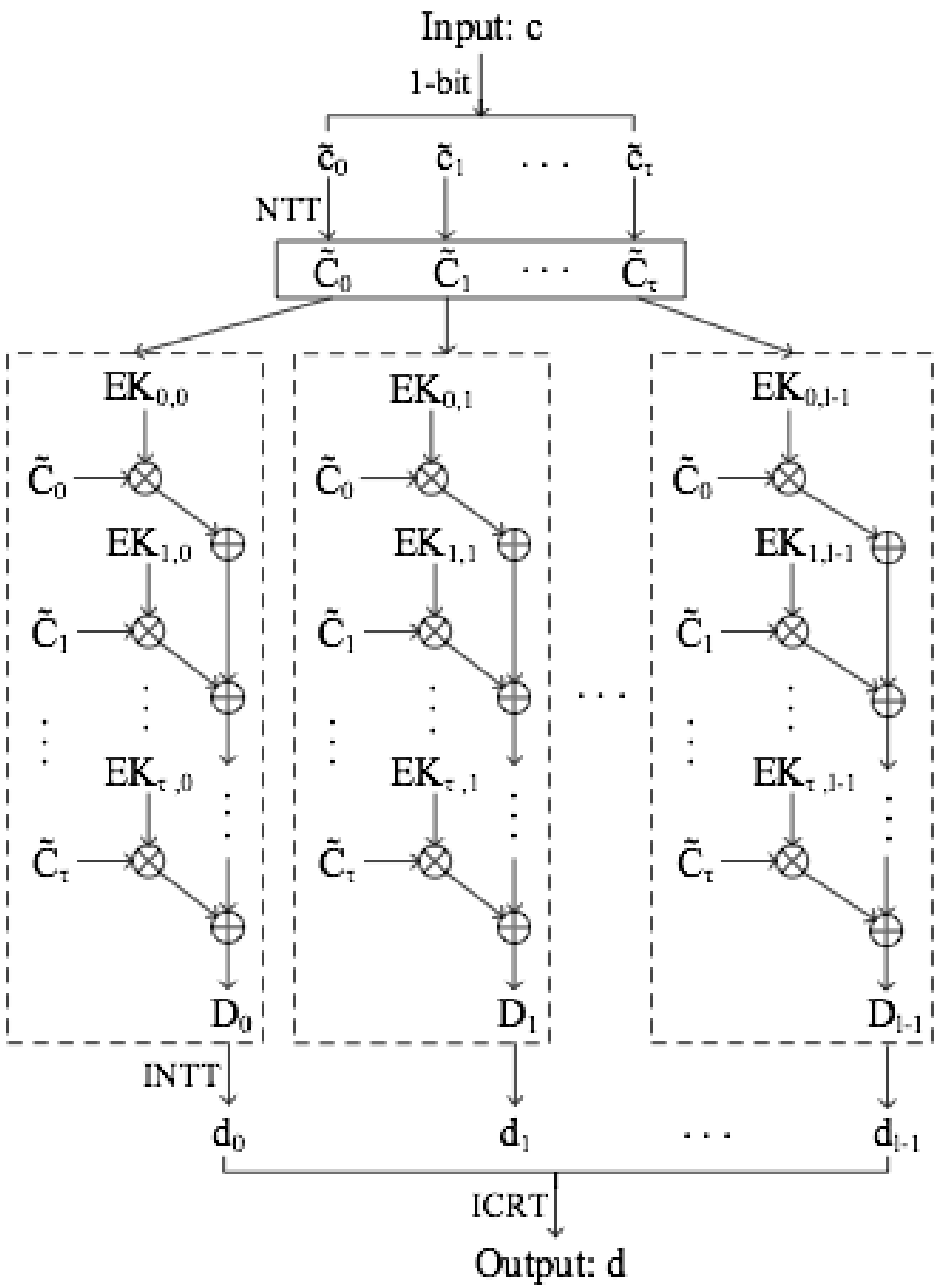
# GPU NTRU: Relinearization

- Evaluation keys are stored in NTT domain
- Computations are mainly in NTT domain

**Algorithm 2** Relinearization

**Input:** Polynomial $c$ with $(n, \log(q))$
**Output:** Polynomial $d$ with $(2n, \log(nq\log(q)))$

1: $\{\tilde{C}_\tau\} = \mathsf{NTT}(\{\tilde{c}_\tau\})$
2: **for** $i = 0 \ldots, l-1$ **do**
3: load $EK_{i,0}, EK_{i,1}, \cdots, EK_{i,\lceil \log(q) \rceil - 1}$
4: $\{D_i\} = \{\sum_{\tau=0}^{\lceil \log(q) \rceil - 1} \tilde{C}_\tau \cdot EK_{i,\tau}\}$
5: **end for**
6: $\{d_i\} = \mathsf{INTT}(\{D_i\})$
7: $d = \mathsf{ICRT}(\{d_i\})$




- EKs are huge (23 GB)
- Memory copy takes most of time
- Page-locked host memory
- Prime numbers:

$$P > \lceil \log(q) \rceil \cdot n \cdot p_i$$

$$\prod_{i=0}^{l-1} p_i > \lceil \log(q) \rceil \cdot n \cdot q$$

# GPU NTRU

- NTL Data $\longleftrightarrow$ 1-D arrays
- Coefficient and polynomial reductions

# Implementation

Implementation Parameters

| | AES | PRINCE |
|---|---|---|
| **Levels** | 40 | 24 |
| **Polynomial Size** | (32768, 1271) | (16384, 575) |
| **Maximum Size of Evaluation Keys** | 23 GBytes | 2 GBytes |

**Server specs:**

- Intel Xeon E5-2609
  @2.5 GHz, 64 GB (1 thread)
- NVIDIA GeForce GTX690
  @915 MHz, 3072 CUDA cores, 4 GB (1536 cores, 2 GB)

# GPU NTRU

TABLE II. TIMING COMPARISON BETWEEN THE CPU AND GPU IMPLEMENTATIONS FOR THE OPERATIONS.

| | Prince | | | AES | | |
|---|---|---|---|---|---|---|
| | GPU | CPU | SPEEDUP | GPU | CPU | SPEEDUP |
| MULTIPLICATION | 0.063 | 0.18 | $\times$ **2.8** | 0.34 | 0.97 | $\times$**2.8** |
| RELINEARIZATION | 0.89 | 10.9 | $\times$**12.2** | 8.97 | 103.37 | $\times$**11.5** |

TABLE III. PERFORMANCE COMPARISON OF PRINCE AND AES IMPLEMENTATIONS.

| | | TOTAL TIME | #BLOCKS | PER BLOCK | Speedup |
|---|---|---|---|---|---|
| **AES** | SIMD Xeon [9] | 36 h | 54 | 2400 sec | $\times$1 |
| | Byte Xeon [9] | 65 h | 720 | 300 sec | $\times$8 |
| | NTRU Xeon [10] | 31 h | 2048 | 55 sec | $\times$43 |
| | **Ours (GPU)** | **4.15 h** | **2048** | **7.3 sec** | $\times$**328** |
| **Prince** | Prince [32] | 57 min | 1024 | 3.3 sec | $\times$1 |
| | **Ours (GPU)** | **22 min** | **1024** | **1.28 sec** | $\times$**2.57** |

# Questions?

# Thank you.