Exploring the Advantages and Challenges of Fermat NTT in FHE Acceleration

CRYPTO 2024

Andrey Kim*†, Ahmet Can Mert, Anisha Mukherjee, **Aikata Aikata**, Maxim Deryabin*, Sunmin Kwon*, Hyung Chul Kang*, Sujoy Sinha Roy

IAIK – Graz University of Technology, Austria
* Samsung Advanced Institute of Technology, Suwon, Republic of Korea; † Altbridge, Inc.

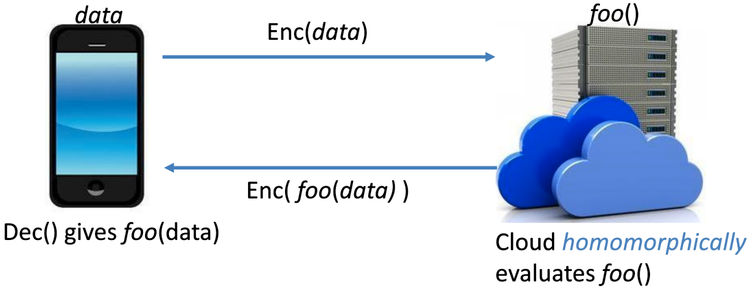
Fully Homomorphic Encryption

Fully Homomorphic Encryption

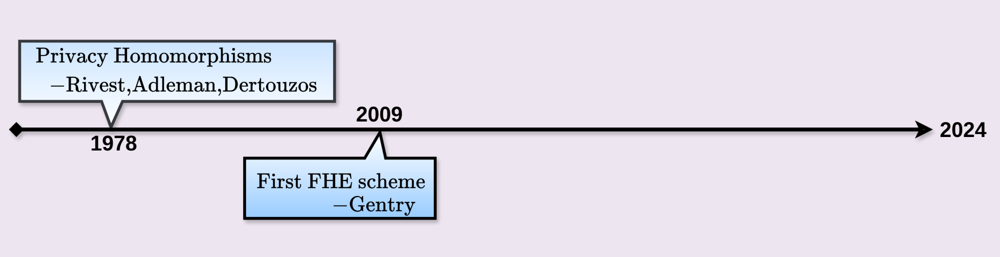
- Fully Homomorphic Encryption (FHE) allows computation on encrypted data.

Fully Homomorphic Encryption

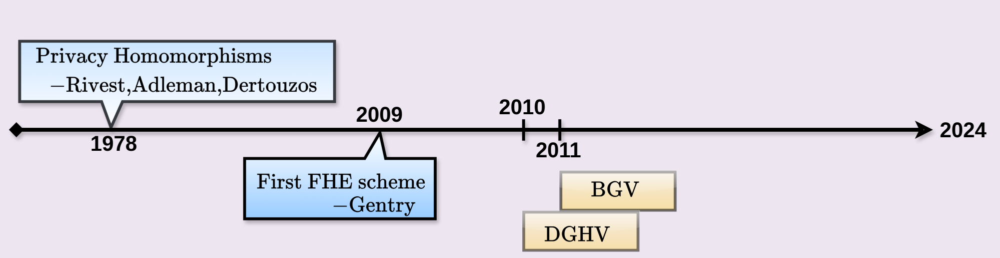
- Fully Homomorphic Encryption (FHE) allows computation on encrypted data.
- Enables secure processing of sensitive data in untrusted environments.



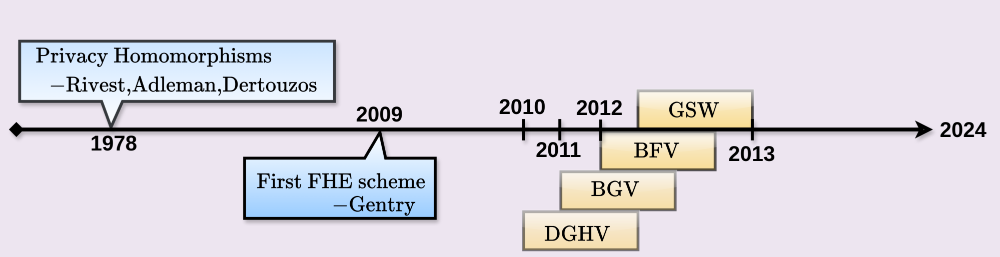
Fully Homomorphic Encryption



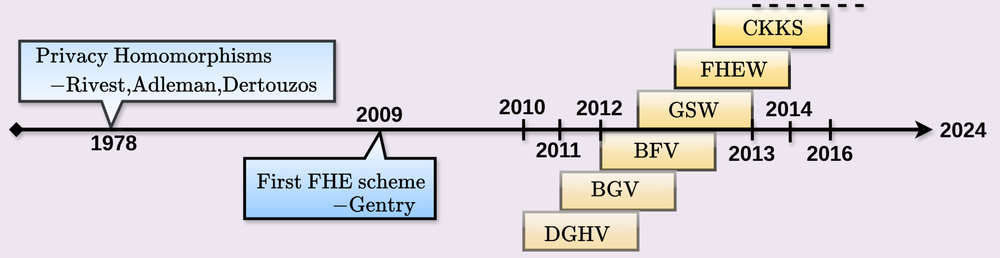
Fully Homomorphic Encryption



Fully Homomorphic Encryption



Fully Homomorphic Encryption



Fully Homomorphic Encryption

- FHE schemes are computationally intensive.

Fully Homomorphic Encryption

- FHE schemes are computationally intensive. 10,000-100,000$\times$ plain comp.

Fully Homomorphic Encryption

- FHE schemes are computationally intensive. 10,000-100,000$\times$ plain comp.

Solution: Hardware Acceleration.

Fully Homomorphic Encryption

- FHE schemes are computationally intensive. 10,000-100,000$\times$ plain comp.

Solution: Hardware Acceleration.

SHARP: A Short-Word Hierarchical Accelerator for Robust and Practical Fully Homomorphic Encryption

Jongmin Kim, Seoul National University, Seoul, Republic of, jongmin.kim@sn
Sangpyo Kim, Seoul National University
Jaewan Choi, Seoul National University
Jaiyoung Pa, Seoul National Uni, Seoul, Republic of, jeff1273@snu.a

REED: Chiplet-based Accelerator for Fully Homomorphic Encryption

Aikata Aikata[1], Ahmet Can Mert[1], Sunmin Kwon[2], Maxim Deryabin[2], Sujoy Sinha Roy[1]
, Graz University of Technology, [2]Samsung Advanced Institute of Technology, Suwon, Republic of Korea

HEAX: An Architecture for Computing on Encrypted Data

M. Sadegh Riazi, UC San Diego, mriazi@ucsd.edu
Kim Laine, Microsoft Research, kim.laine@microsoft.com
Blake Pelton, Microsoft, blakep@microsoft.com
Wei Dai, Microsoft Research, wei.dai@microsoft.com

CraterLake: A Hardware Accelerator for Efficient Unbounded Computation on Encrypted Data

Nikola Samardzic, nsamar@csail.mit.edu, Massachusetts Institute of Technology, Cambridge, MA, USA
Axel Feldmann, axelf@csail.mit.edu, Massachusetts Institute of Technology, Cambridge, MA, USA
Aleksandar Krastev, alexalex@csail.mit.edu, Massachusetts Institute of Technology, Cambridge, MA, USA

BASALISC: Programmable Hardware Accelerator for BGV Fully Homomorphic Encryption

Distribution Statement A: Approved for Public Release, Distribution Unlimited

Robin Geelen[1]*, Michiel Van Beirendonck[1]*, Hilder V. L. Pereira[1], Brian Huffman[2], Tynan McAuley[3], Ben Selfridge[2], Daniel Wagner[2], Georgios Dimou[3], Ingrid Verbauwhede[1], Frederik Vercauteren[1] and David W. Archer[2]

FPT: a Fixed-Point Accelerator for Torus Fully Homomorphic Encryption

Michiel Van Beirendonck, michiel.vanbeirendonck@esat.kuleuven.be, COSIC, KU Leuven, Leuven, Belgium
Jan-Pieter D'Anvers, janpieter.danvers@esat.kuleuven.be, COSIC, KU Leuven, Leuven, Belgium
Furkan Turan, furkan.turan@esat.kuleuven.be, COSIC, KU Leuven, Leuven, Belgium
Ingrid Verbauwhede, ingrid.verbauwhede@esat.kuleuven.be, COSIC, KU Leuven, Leuven, Belgium

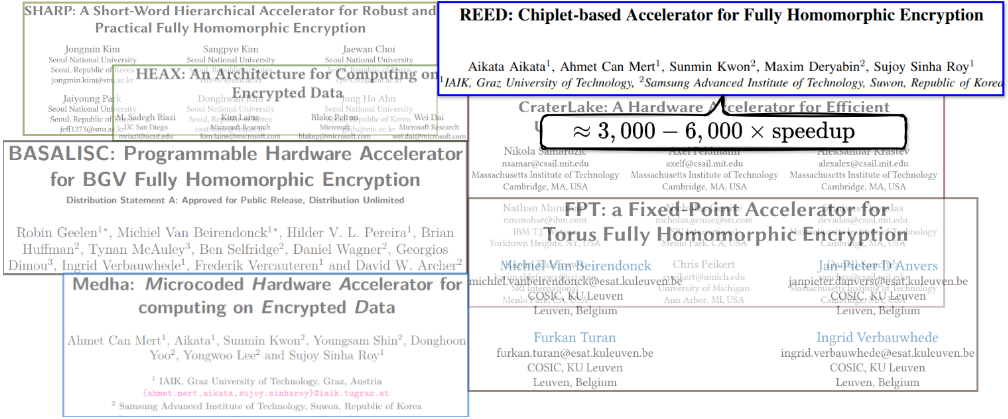
Medha: Microcoded Hardware Accelerator for computing on Encrypted Data

Ahmet Can Mert[1], Aikata[1], Sunmin Kwon[2], Youngsam Shin[2], Donghoon Yoo[2], Yongwoo Lee[2] and Sujoy Sinha Roy[1]
[1] IAIK, Graz University of Technology, Graz, Austria
{ahmet.mert,aikata,sujoy.sinharoy}@iaik.tugraz.at
[2] Samsung Advanced Institute of Technology, Suwon, Republic of Korea

Fully Homomorphic Encryption

- FHE schemes are computationally intensive. 10,000-100,000$\times$ plain comp.

Solution: Hardware Acceleration.



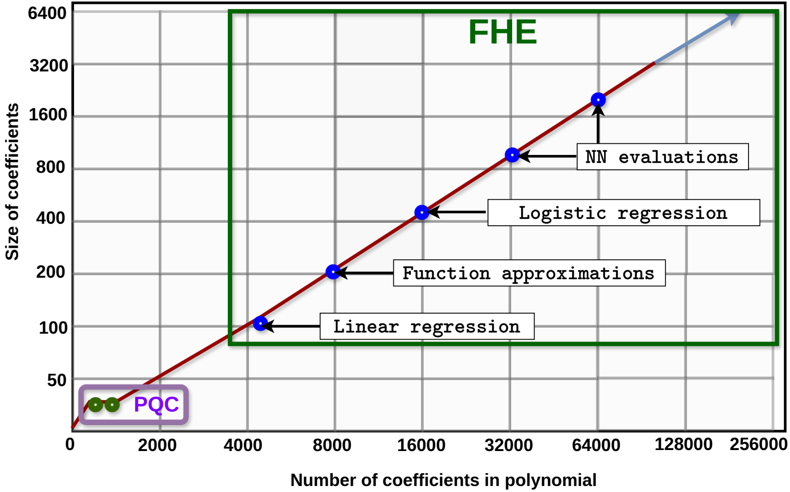
Acceleration Challenges

Challenges in accelerating Homomorphic Encryption

Many polynomial arithmetic operations

- Large degree polynomial arithmetic
- Long integer arithmetic

Challenges with practical realization of Homomorphic Encryption

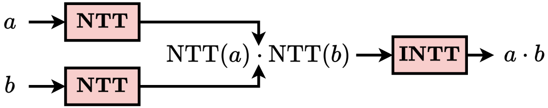


Challenges in accelerating Homomorphic Encryption

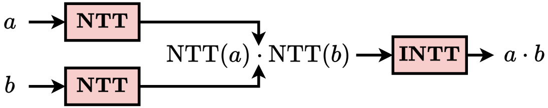
Many polynomial arithmetic operations-**Handled using NTT/INTT unit**

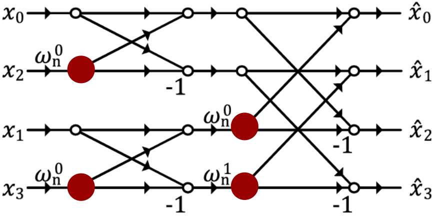
- Large degree polynomial arithmetic
- Long integer arithmetic

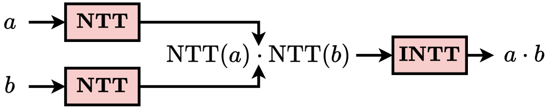
NTT Transformation

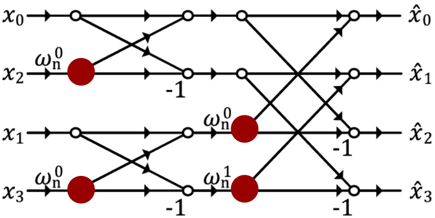


NTT Transformation



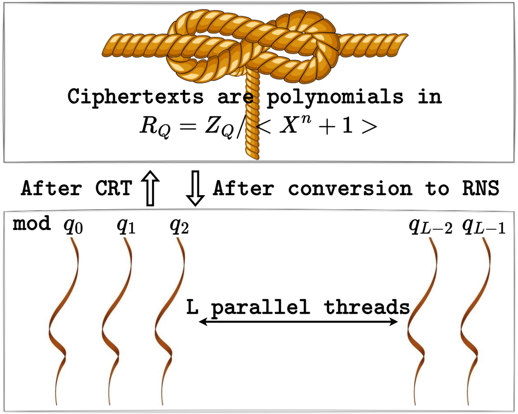




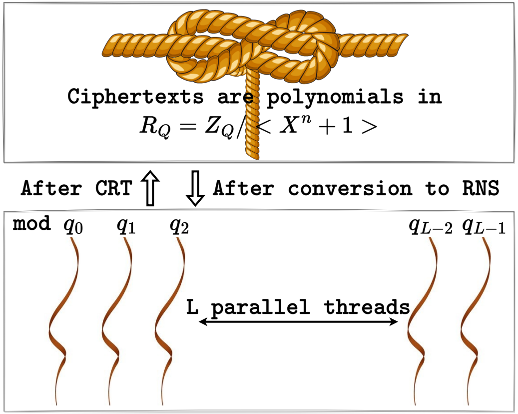


NTT transformation for 1600-bit Q is very expensive.

Residue Number System



Residue Number System



One NTT transformation is cheap, but it has to support multiple moduli.

Prior Works

REED – Arxiv'23

Components	28nm (mm^2)		7nm (mm^2)	
	1024×64	**512×128**	**1024×64**	**512×128**
REED	74.9	115	24	43.9
REED-PU	58.0	81.0	7.01	9.9
⌊ NTT/INTT	38.2	56.8	5.61	7.9
⌊ 2×MAS	3.1	6.6	0.42	0.76
⌊ PRNG	0.15	0.28	0.02	0.04
⌊ 2×AUT	0.14	0.32	0.02	0.04
⌊ Memory	16.1	16.1	1.2	1.2

F1 – MICRO'21

Component	Area [mm^2]	TDP [W]
NTT FU	2.27	4.80
Automorphism FU	0.58	0.99
Multiply FU	0.25	0.60
Add FU	0.03	0.05
Vector RegFile (512 KB)	0.56	1.67
Compute cluster	3.97	8.75
(NTT, Aut, 2× Mul, 2× Add, RF)		
Total compute (16 clusters)	**63.52**	**140.0**

ARK – ISCA'22

Component	**Area (mm^2)**	**Peak power (W)**
4 BConvUs	9.3	18.9
4 NTTUs	57.2	95.2
4 AutoUs	20.6	4.6
8 MADUs	8.9	24.7

CraterLake – ISCA'22

Component	Area [mm^2]
CRB FU	158.8
NTT FU	28.1
Automorphism FU	9.0
KSHGen FU	3.3
Multiply FU	2.2
Add FU	0.8
Total FUs (CRB, 2×NTT, Aut, KSHGen, 5×Mul, 5×Add)	**240.5**

Problem Statement

Can we make modular multiplications in the NTT/INTT units extremely **inexpensive** *and ensure NTT* **reusability***?*

The Fermat-number based Technique

The Fermat-number based Technique

The FNTT method:

Use the Fermat number $(\mathrm{P} = 2^{K} + 1)$ as an *auxiliary* modulus before NTT.

The Fermat-number based Technique

The FNTT method:

Use the Fermat number $(\mathrm{P} = 2^K + 1)$ as an *auxiliary* modulus before NTT.

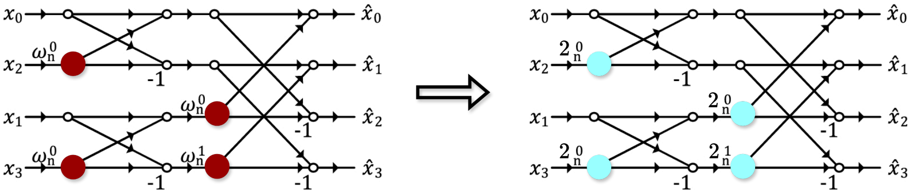
$$
\begin{tikzcd}
\mathbb{Z}_{q_i}[X] \arrow[r, "id"] \arrow[d, dashed, "\times"] \arrow[dr, phantom, "\circlearrowleft"] & \mathcal{R}_{q_i} \arrow[r, "\mathsf{NTT}_{\mathsf{q_i}}"] \arrow[d, dashed, "\times"] \arrow[dr, phantom, "\circlearrowleft"] & \{\mathbb{Z}_{q_i}\}_N \arrow[d, "\odot"] \\
\mathbb{Z}_{q_i}[X] & \mathcal{R}_{q_i} \arrow[l, hook', "id"] & \{\mathbb{Z}_{q_i}\}_N \arrow[l, "\mathsf{INTT}_{\mathsf{q_i}}"]
\end{tikzcd}
$$

The FNTT method:

Use the Fermat number $(\mathrm{P} = 2^{K} + 1)$ as an *auxiliary* modulus before NTT.

$$
\begin{tikzcd}
\mathbb{Z}_{q_i}[X] \arrow[r, "id"] \arrow[d, dashed, "\times"] \arrow[dr, phantom, "\circlearrowleft"] & \mathcal{R}_{q_i} \arrow[r, "\mathsf{emb}"] \arrow[d, dashed, "\times"] \arrow[dr, phantom, "\circlearrowleft"] & \mathcal{R}_P \arrow[r, "\mathsf{NTT_P}"] \arrow[d, dashed, "\times"] \arrow[dr, phantom, "\circlearrowleft"] & \{\mathbb{Z}_P\}_N \arrow[d, "\odot"] \\
\mathbb{Z}_{q_i}[X] & \mathcal{R}_{q_i} \arrow[l, "id"] & \mathcal{R}_P \arrow[l, "(\mathrm{mod}\ \mathsf{q_i})"] & \{\mathbb{Z}_P\}_N \arrow[l, "\mathsf{INTT_P}"]
\end{tikzcd}
$$

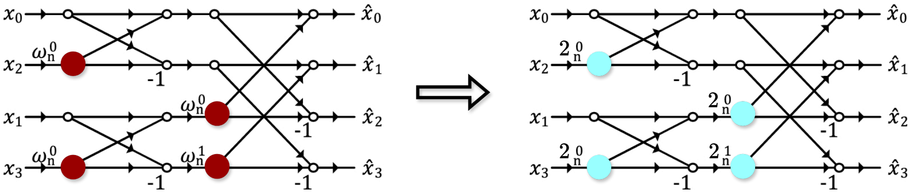
The Fermat-number based Technique



Advantages:

- All modular multiplications during NTT/INTT are transformed into simple shift operations. $\rightarrow$ Multiplier-less NTT

The Fermat-number based Technique



Advantages:

- All modular multiplications during NTT/INTT are transformed into simple shift operations. $\rightarrow$ Multiplier-less NTT
- The roots of unity modulo P are powers-of-two. $\rightarrow$ No storage required

The Fermat-number based Technique

Challenge:

- $\mathrm{P} = 2^K + 1$ K has K power-of-two twiddle factors.
- With K-th root of unity, we can perform $\frac{K}{2}$-point negacyclic NTT.

The Fermat-number based Technique

Challenge:

- $\mathrm{P} = 2^K + 1$ K has K power-of-two twiddle factors.
- With K-th root of unity, we can perform $\frac{K}{2}$-point negacyclic NTT.

Example: For $N = 2^{16}$, required $K = 2^{16} \rightarrow \mathrm{P} = 2^{2^{16}} + 1$. ($65, 536$-bit large modulus)

The Fermat-number based Technique

Challenge:

- $\mathrm{P} = 2^K + 1$ K has K power-of-two twiddle factors.
- With K-th root of unity, we can perform $\frac{K}{2}$-point negacyclic NTT.

Example: For $N = 2^{16}$, required $K = 2^{16} \rightarrow \mathrm{P} = 2^{2^{16}} + 1$. ($65,536$-bit large modulus)
The auxiliary modulus is $1,214\times$ larger than actual moduli q_i (e.g., 54-bit).

Solution: We propose utilizing univariate polynomial to multivariate switch.

Solution: We propose utilizing univariate polynomial to multivariate switch.

$$\begin{tikzcd}
\mathbb{Z}_{q_i}[X] \arrow[r, "id"] \arrow[d, dashed, "\times"] \arrow[dr, phantom, "\circlearrowleft"] & \mathcal{R}_{q_i} \arrow[r, "\text{emb}"] \arrow[d, dashed, "\times"] \arrow[dr, phantom, "\circlearrowleft"] & \mathcal{R}_P \arrow[r, "\mathsf{NTT_P}"] \arrow[d, dashed, "\times"] \arrow[dr, phantom, "\circlearrowleft"] & \{\mathbb{Z}_P\}_N \arrow[d, "\odot"] \\
\mathbb{Z}_{q_i}[X] & \mathcal{R}_{q_i} \arrow[l, "id"'] & \mathcal{R}_P \arrow[l, "(\text{mod } \mathsf{q_i})"'] & \{\mathbb{Z}_P\}_N \arrow[l, "\mathsf{INTT_P}"']
\end{tikzcd}$$

Solution: We propose utilizing univariate polynomial to multivariate switch.

$$\begin{tikzcd}
\mathbb{Z}_{q_i}[X] \arrow[r, "id"] \arrow[d, dashed, "\times"] \arrow[dr, phantom, "\circlearrowleft"] & \mathcal{R}_{q_i} \arrow[r, "\mathsf{emb}"] \arrow[d, dashed, "\times"] \arrow[dr, phantom, "\circlearrowleft"] & \mathcal{R}_P \arrow[r, "\mathsf{emb}"] \arrow[d, dashed, "\times"] \arrow[dr, phantom, "\circlearrowleft"] & \mathcal{S}_P \arrow[r, "\mathsf{NTT_P}"] \arrow[d, dashed, "\times"] \arrow[dr, phantom, "\circlearrowleft"] & \left\{\{\mathbb{Z}_P\}_{K_i}\right\}_{i=1}^{\ell} \arrow[d, "\odot"] \\
\mathbb{Z}_{q_i}[X] & \mathcal{R}_{q_i} \arrow[l, "id"'] & \mathcal{R}_P \arrow[l, "(\mathrm{mod}\ \mathsf{q_i})"'] & \mathcal{S}_P \arrow[l, "\mathsf{revert}"'] & \left\{\{\mathbb{Z}_P\}_{K_i}\right\}_{i=1}^{\ell} \arrow[l, "\mathsf{INTT_P}"']
\end{tikzcd}$$

The Fermat-number based Technique

Our technique: For $N = 2^{16}$, we take required $K = 2^7 \rightarrow \mathrm{P} = 2^{2^7} + 1$.

The Fermat-number based Technique

Our technique: For $N = 2^{16}$, we take required $K = 2^7 \rightarrow \mathrm{P} = 2^{2^7} + 1$.

- With K-th root of unity, we can perform $\frac{K}{2}$-point negacyclic NTT.

The Fermat-number based Technique

Our technique: For $N = 2^{16}$, we take required $K = 2^7 \rightarrow \mathrm{P} = 2^{2^7} + 1$.

■ With K-th root of unity, we can perform $\frac{K}{2}$-point negacyclic NTT.

We break $\mathcal{R}_q = \mathbb{Z}_q/(X^{2^{16}} + 1)$ as follows:

$$
\begin{aligned}
&\Rightarrow (X^{2^{16}} + 1) \\
&\Rightarrow (X^{2^6} + 1) \times (X^{2^{10}} + 1) \\
&\Rightarrow (X^{2^6} + 1) \times (X^{2^6} + 1) \times (X^{2^4} + 1)
\end{aligned}
$$

The Fermat-number based Technique

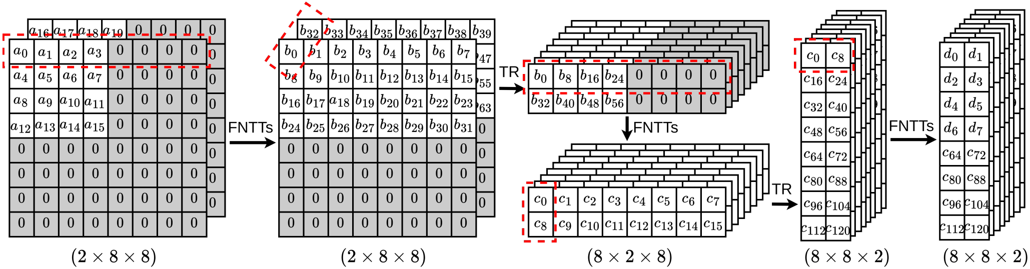
Toy Example: For $N = 2^5$, and $K = 2^4 \rightarrow \mathrm{P} = 2^{2^4} + 1$.

$$\mathcal{R}_q = \mathbb{Z}_q/(X^{2^5} + 1) \quad \Rightarrow \quad (X^{2^2} + 1) \times (X^{2^2} + 1) \times (X^{2^1} + 1)$$

The Fermat-number based Technique

Toy Example: For $N = 2^5$, and $K = 2^4 \rightarrow \mathrm{P} = 2^{2^4} + 1$.

$$\mathcal{R}_q = \mathbb{Z}_q/(X^{2^5}+1) \quad \Rightarrow \quad (X^{2^2}+1) \times (X^{2^2}+1) \times (X^{2^1}+1)$$



The Fermat-number based Technique

Implications:

- The auxiliary modulus is only $2\times$ larger than actual moduli q_i (e.g., 54-bit).
- The number of coefficients increase by $4\times$.

The Fermat-number based Technique

Implications:

- The auxiliary modulus is only $2\times$ larger than actual moduli q_i (e.g., 54-bit).
- The number of coefficients increase by $4\times$.

The multivariate has $8\times$ more data overhead than the prior $1,214\times$ overhead.

The Implementation Methodology

The Implementation Methodology

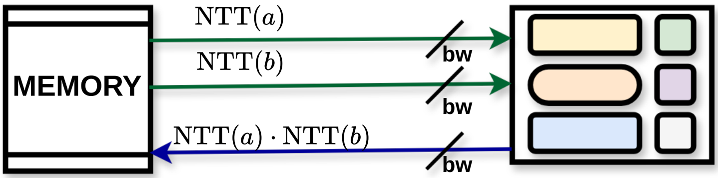
Building Blocks of an FHE accelerator:

- NTT/INTT Unit
- Multiply-and-Accumulate Unit
- Automorphism/Conjugation Unit

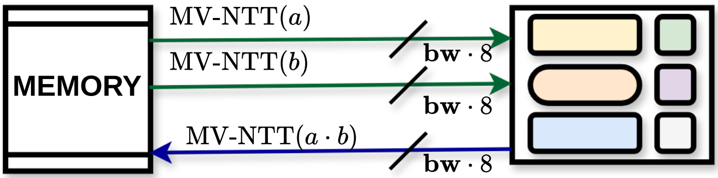
The Implementation Methodology

The Communication vs Computation trade-off: Data in FNTT form is $8\times$ larger than data in prior NTT forms.

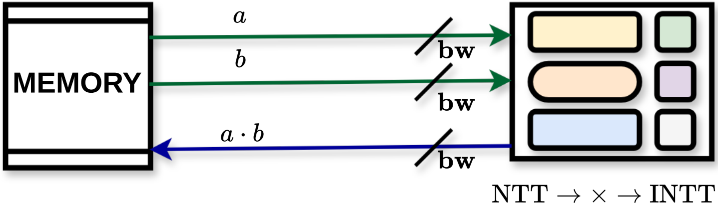
The Communication vs Computation trade-off: Data in FNTT form is $8\times$ larger than data in prior NTT forms.



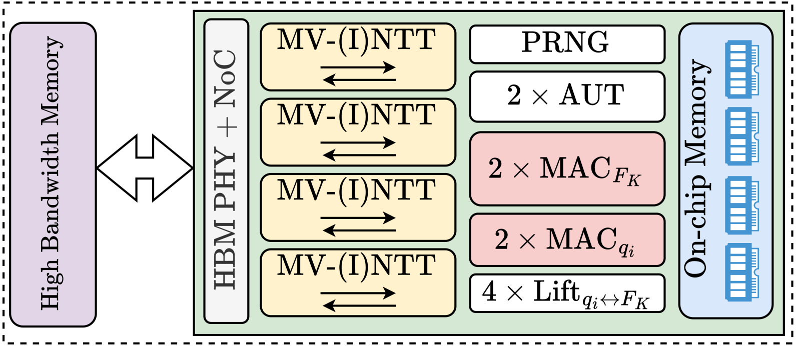
The Communication vs Computation trade-off: Data in FNTT form is $8\times$ larger than data in prior NTT forms.



The Communication vs Computation trade-off: Data in FNTT form is $8\times$ larger than data in prior NTT forms.



To support increased MV-NTT/MV-INTT overhead, we instantiate multiple units.



Implementation Results

The results are for the 28nm ASIC technology, and our design runs at 1.2GHz.

Building Block	**Area** (mm^2)
MV-NTT/INTT	4×21.5 (vs. 38.2mm^2 [REED])
MV-NTT/INTT (mem.)	4×17.4
MAC (for F_K and q_i)	2×2.8
Automorphism	2×0.14
On-chip memory	52
HBM3 PHY+NoC	33.8
Total	250.94

We achieve $1,200\times$ speed-up compared to software implementation.

Future Scope

Limitations and Future Scope

- **Communication Overhead:** Multiplierless NTTs are cheap, but having to instantiate multiple such units mitigates this advantage.
  - The proposed MV-NTT finds potential in the emerging PIM architectures.

Limitations and Future Scope

- **Communication Overhead:** Multiplierless NTTs are cheap, but having to instantiate multiple such units mitigates this advantage.
  - The proposed MV-NTT finds potential in the emerging PIM architectures.
- **Computation Overhead:** The amount of arithmetic computation required grows proportional to the reduced multiplications.
  - Implementation of Schönhage-Strassen or Nussbaumer Approach should be explored as they theoretically reduce the operation count.

Exploring the Advantages and Challenges of Fermat NTT in FHE Acceleration

CRYPTO 2024

Andrey Kim*†, Ahmet Can Mert, Anisha Mukherjee, **Aikata Aikata**, Maxim Deryabin*, Sunmin Kwon*, Hyung Chul Kang*, Sujoy Sinha Roy

IAIK – Graz University of Technology, Austria
* Samsung Advanced Institute of Technology, Suwon, Republic of Korea; † Altbridge, Inc.