FHE for hardware, hardware for FHE and beyond!

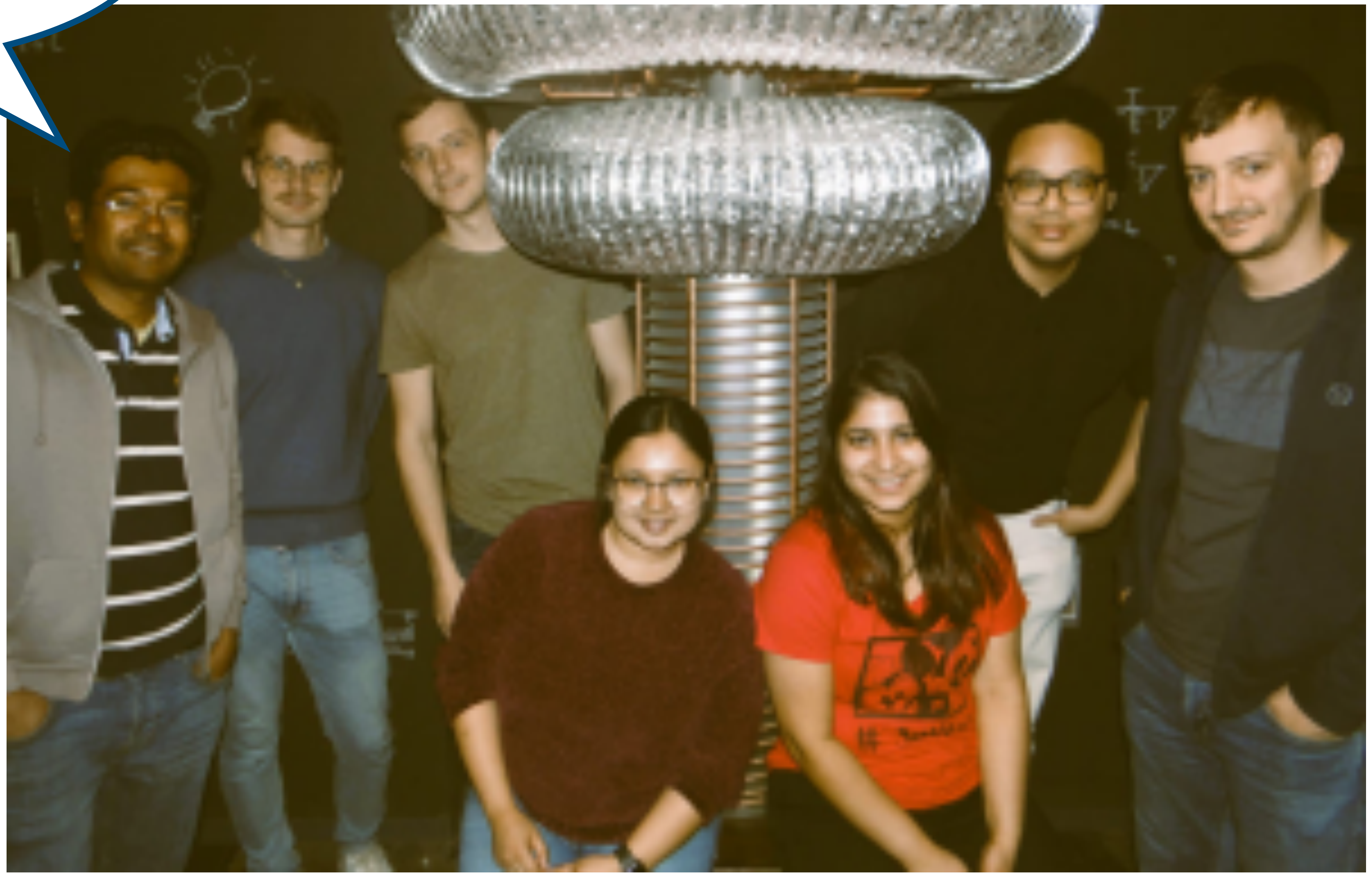
Anisha Mukherjee

IAIK, Graz University of Technology, Austria

September 4, 2024

Our Team

Team Sinha Roy



- HW and design for homomorphic encryption
- Post-quantum cryptographic schemes
- Zero-knowledge proofs

Outline

- **Background and Motivation**
  - Homomorphic Encryption (HE)
  - Ring-LWE based HE and challenges
- **FHE-HW: Hardware acceleration for HE**
  - FNTT: Fermat's Number Technique for NTT
  - REED: Chiplet-based hardware accelerator
- **HW-FHE: HE for hardware reusability**
  - ModHE: Module-LWE based HE scheme
- **Beyond HE**
  - Hybrid Homomorphic Encryption (HHE)
  - SASTA: Fault attack on HHE

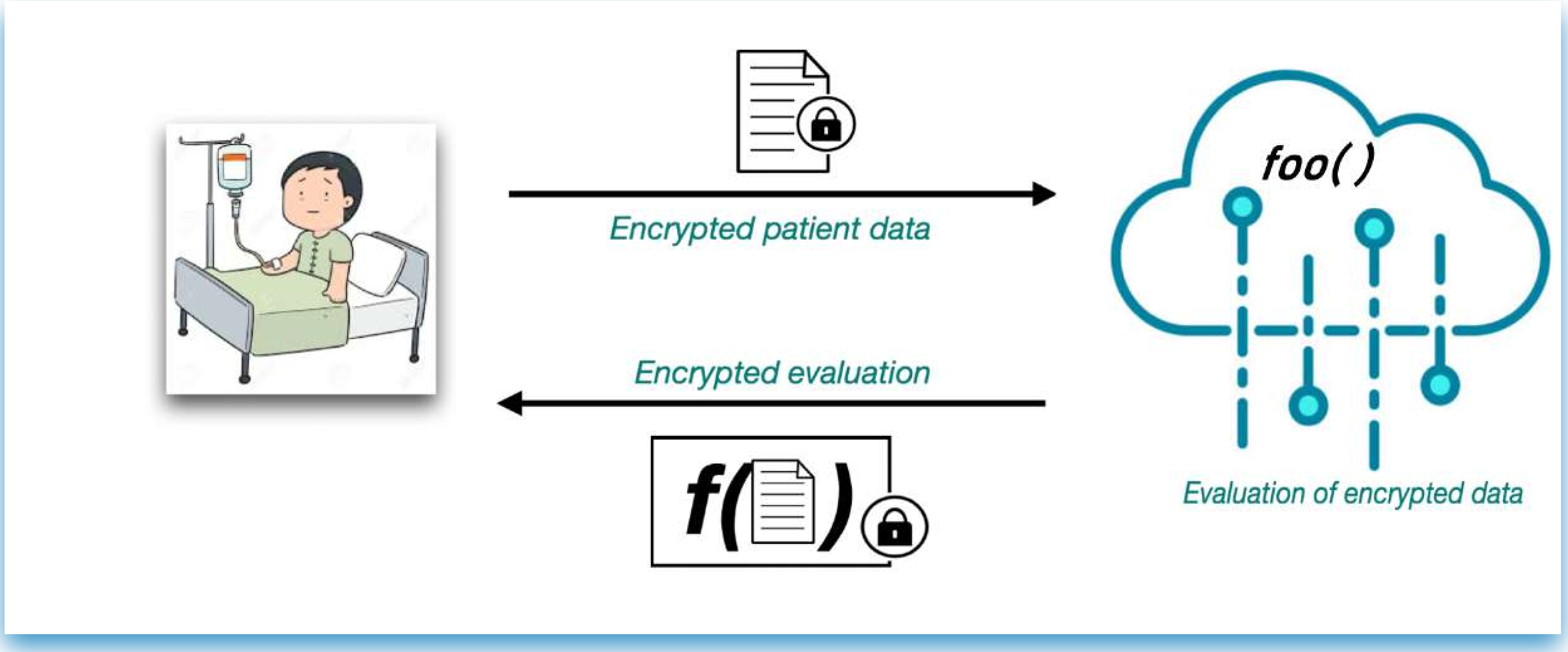
Outline

- **Background and Motivation**
  - Homomorphic Encryption (HE)
  - Ring-LWE based HE and challenges
- **FHE-HW: Hardware acceleration for HE**
  - FNTT: Fermat's Number Technique for NTT
  - REED: Chiplet-based hardware accelerator
- **HW-FHE: HE for hardware reusability**
  - ModHE: Module-LWE based HE scheme
- **Beyond HE**
  - Hybrid Homomorphic Encryption (HHE)
  - SASTA: Fault attack on HHE

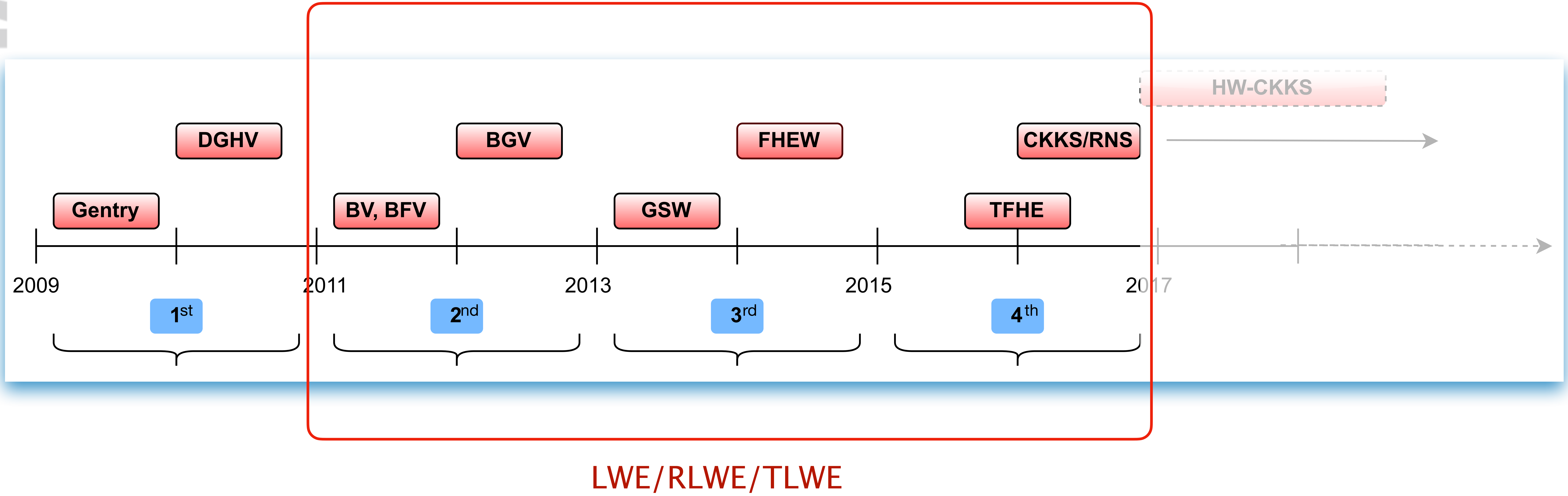
Homomorphic Encryption (HE): Brief introduction

- Allows functional evaluation on encrypted data
- Preserves privacy of data owners

Homomorphic Encryption (HE): Brief introduction



Generations of lattice-based HE



Outline

- **Background and Motivation**
  - Homomorphic Encryption (HE)
  - Ring-LWE based HE and challenges
- **FHE-HW: Hardware acceleration for HE**
  - FNTT: Fermat's Number Technique for NTT
  - REED: Chiplet-based hardware accelerator
- **HW-FHE: HE for hardware reusability**
  - ModHE: Module-LWE based HE scheme
- **Beyond HE**
  - Hybrid Homomorphic Encryption (HHE)
  - SASTA: Fault attack on HHE

Ring-Learning with Errors (RLWE)

$$a(x) \cdot {\color{red}s(x)} + e(x) = b(x) \pmod q \pmod{f(x)}$$

$a(x) = (a_1 x^0 + \cdots + a_N x^{N-1}) \in \mathscr{R}_q$

${\color{red}s(x) \in \mathscr{R}_q}$

$e_i \in \{0, 1, -1\}^N$

$b(x) \in \mathscr{R}_q$

$\mathscr{R}_q = \mathbb{Z}_q[X] / < f(x) >$

$f(x) = x^N + 1$

Eg: $(N = 2^{14}, \ \log q = 411)$

RLWE meets HE 🤝

m 🤝 pk 🔒

$\Longrightarrow$ HE.Enc

$\texttt{ct} = (c_0, c_1) \in \mathscr{R}_q \times \mathscr{R}_q$

$f(\cdot)$ $\Big\Downarrow$ Eval(***m* + *m*', *m* . *m*'**)

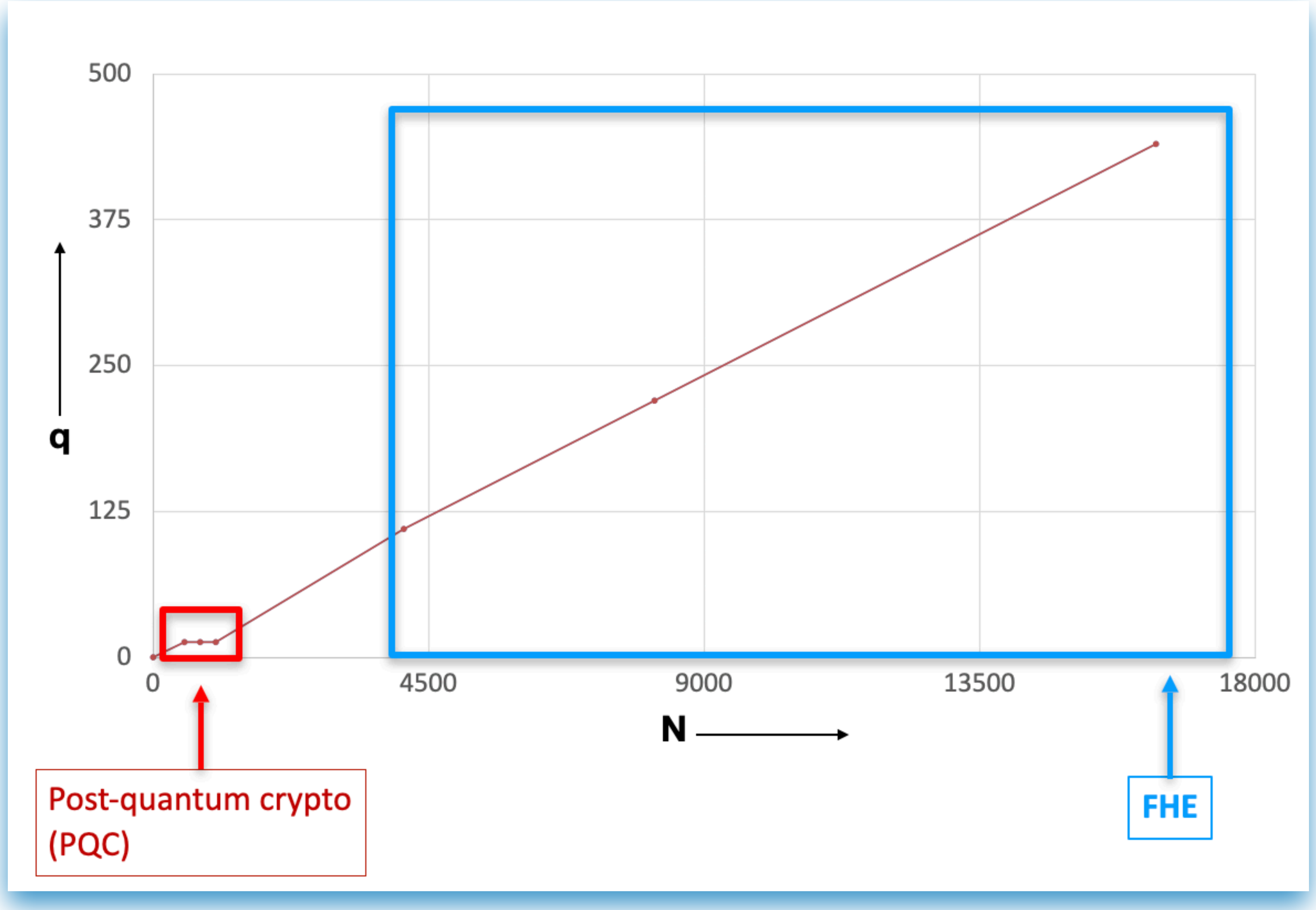
$\texttt{ct}_{\text{add}} = \texttt{ct} + \texttt{ct}' \in \mathscr{R}_q \times \mathscr{R}_q$

$\texttt{ct}_{\text{mult}} = \texttt{ct} \times \texttt{ct}' \in \mathscr{R}_q^3$

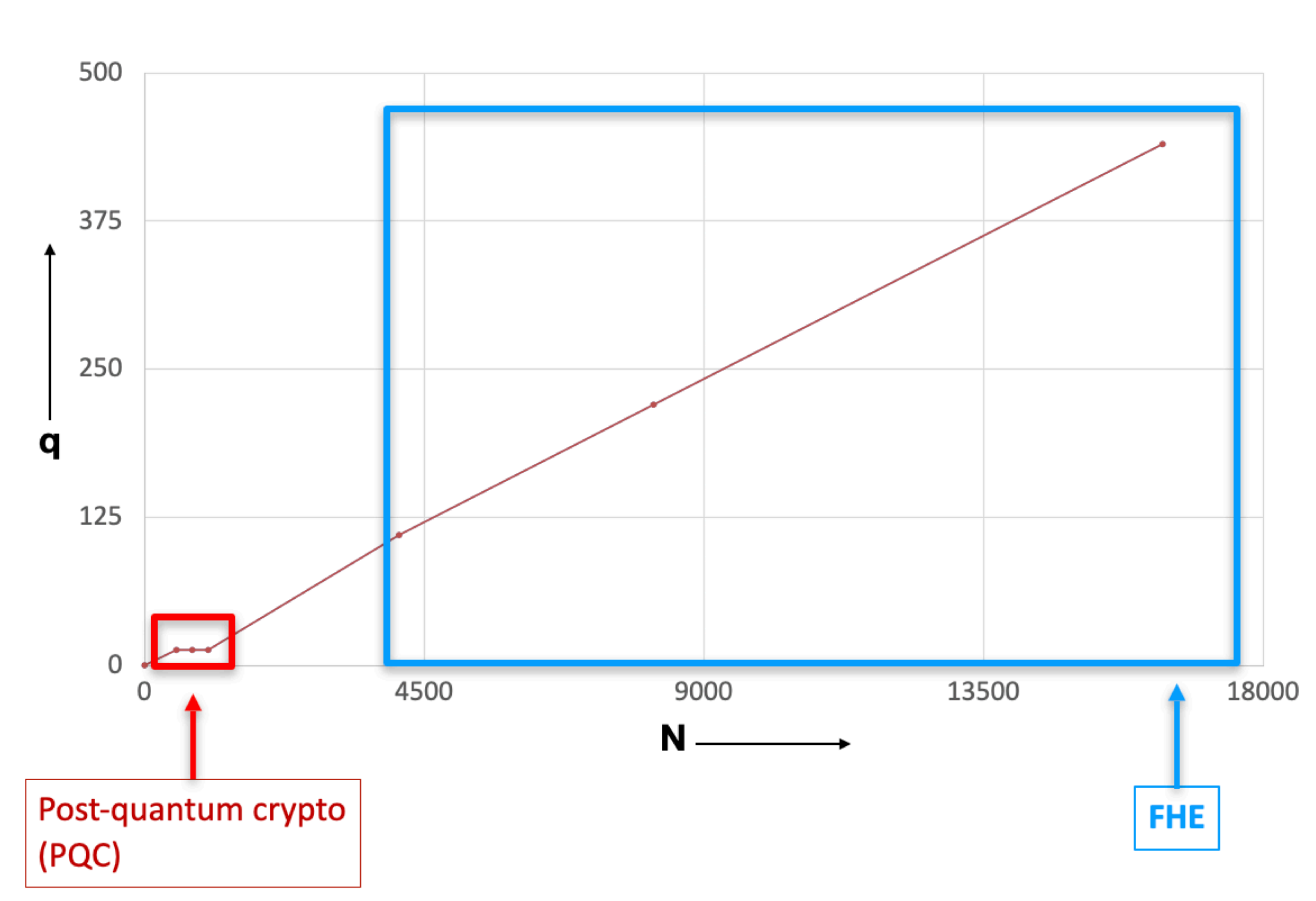
$\Longleftarrow$ HE.Dec

$< \texttt{ct}, s > = c_0 + c_1 \cdot s$

RLWE meets HE: Security and parameter selection



RLWE meets HE: Security and parameter selection



⟹

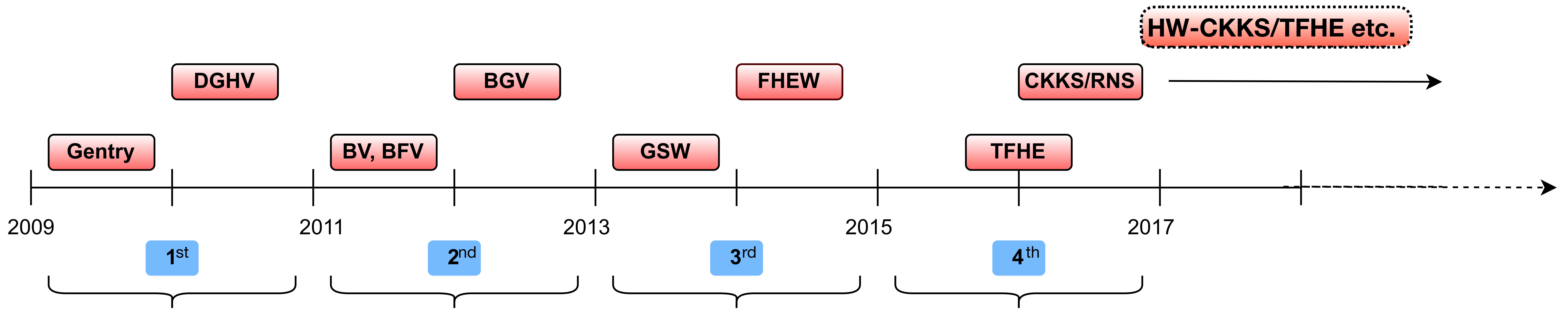
n	security level	logq
1024	128	25
	192	17
	256	13
2048	128	51
	192	35
	256	27
4096	128	101
	192	70
	256	54
8192	128	202
	192	141
	256	109
16384	128	411
	192	284
	256	220
32768	128	827

From: http://homomorphicencryption.org

What's the catch?

- HE schemes are computationally intensive
- Usually incur an overhead of $10^4 - 10^5 \times$ compared to plaintext comp.

Hardware acceleration to close the gap



Challenges with FHE-HW acceleration

- Many (large) polynomial arithmetic operations
  - Large degree polynomial arithmetic
  - Long integer arithmetic
- Memory management
  - Large ciphertext and key sizes
  - Limited on-chip memory

Challenges with FHE-HW acceleration

- Many (large) polynomial arithmetic operations
  - Large degree polynomial arithmetic
  - Long integer arithmetic
- Memory management
  - Large ciphertext and key sizes
  - Limited on-chip memory

- Poly mult is expensive
- Naive method has $\mathcal{O}(N^2)$ complexity
- Use NTT/INTT with $\mathcal{O}(N \log N)$ complexity

NTT : Number Theoretic Transform

Challenges with FHE-HW acceleration

- Many (large) polynomial arit
    - Large degree polynomial
    - Long integer arithmetic

RNS decomposition for efficiency:

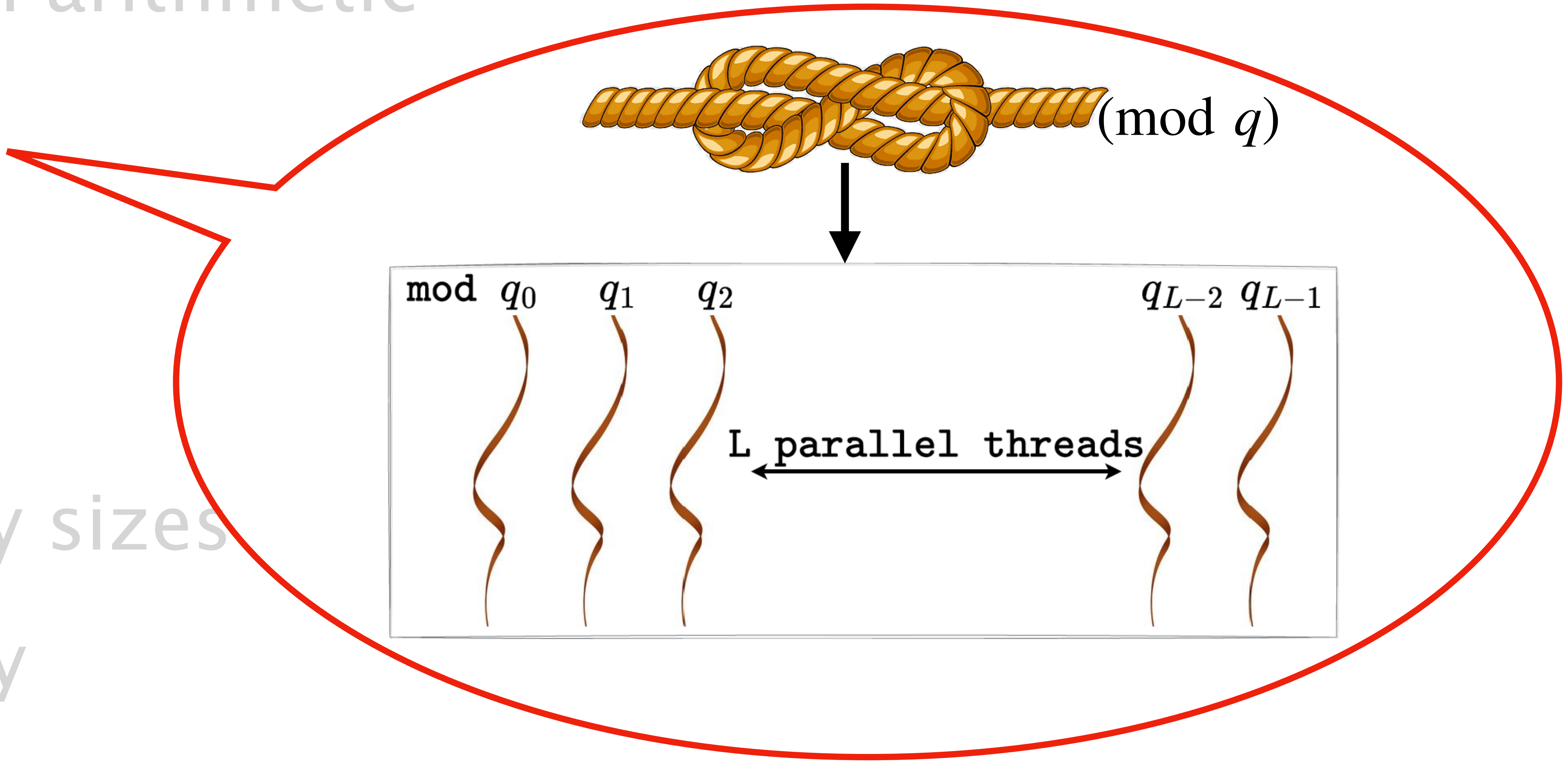
$$q = \prod_i q_i$$

$$\log q_i < q \; \forall i$$

- Memory management
    - Large ciphertext and key sizes
    - Limited on-chip memory

Challenges with FHE-HW acceleration

- Many (large) polynomial arithmetic operations
  - Large degree polynomial arithmetic
  - Long integer arithmetic
- Memory management
  - Large ciphertext and key sizes
  - Limited on-chip memory



Challenges with FHE-HW acceleration

- Many (large) polynomial arithmetic operations
  - NTT/INTT transformation involves modular add, subtract, mult
  - NTT/INTT transformation needs to support multiple RNS moduli
  - FHE requires many such NTT/INTT transformations

Challenges with FHE-HW acceleration

- Many (large) polynomial arithmetic ope
  - NTT/INTT transformation involves
  - NTT/INTT transformation needs to support multiple RNS moduli
  - FHE requires many such NTT/INTT transformations

Fast and resource-efficient modular mult

$\Longrightarrow$high-performance FHE accelerator

FHE-FNTT

Can we make modular multiplications in NTT/INTT units extremely cheap ?

Outline

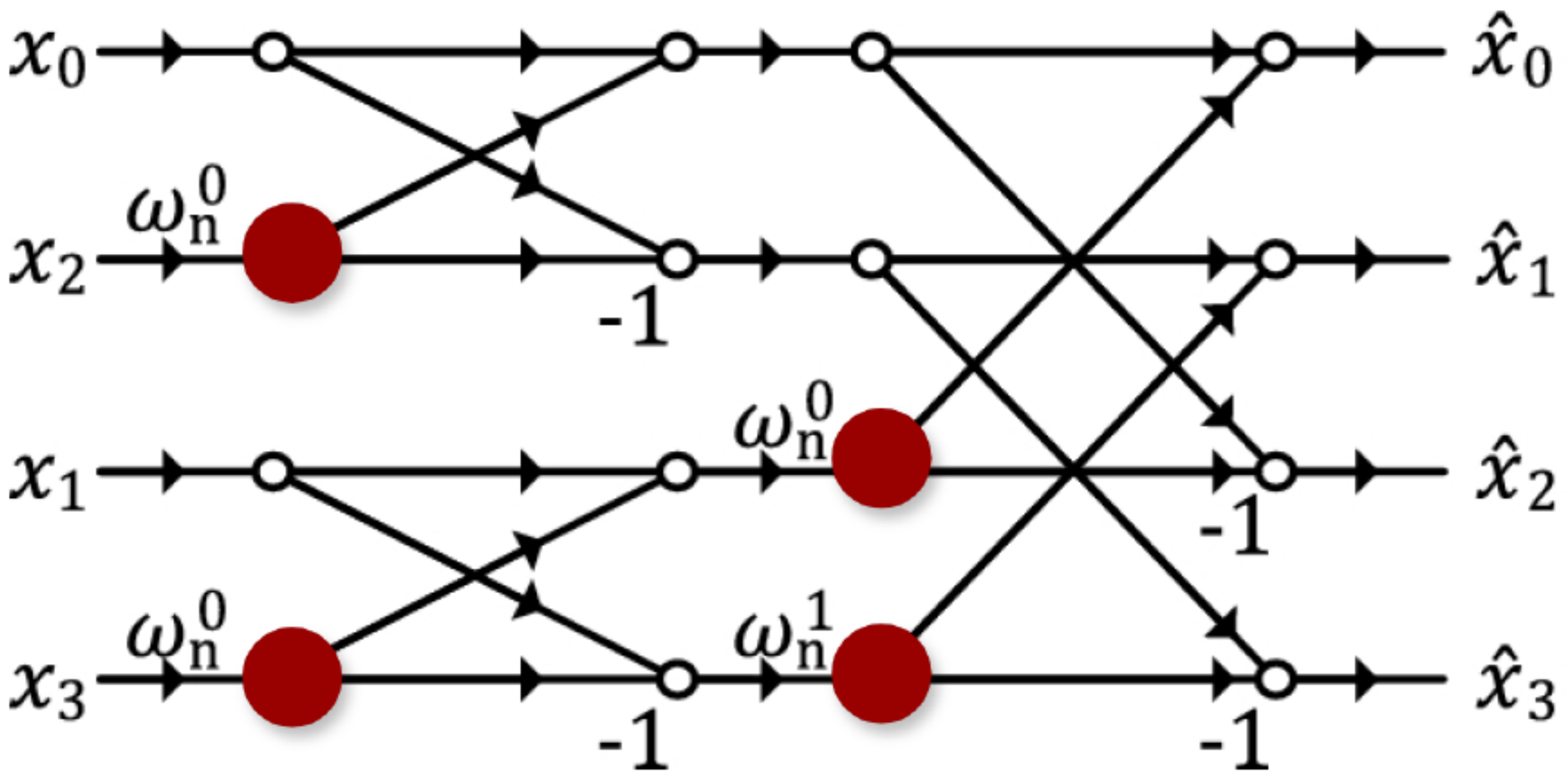
- **Background and Motivation**
  - Homomorphic Encryption (HE)
  - Ring-LWE based HE and challenges
- **FHE-HW: Hardware acceleration for HE**
  - FNTT: Fermat's Number Technique for NTT
  - REED: Chiplet-based hardware accelerator
- **HW-FHE: HE for hardware reusability**
  - ModHE: Module-LWE based HE scheme
- **Beyond HE**
  - Hybrid Homomorphic Encryption (HHE)
  - SASTA: Fault attack on HHE

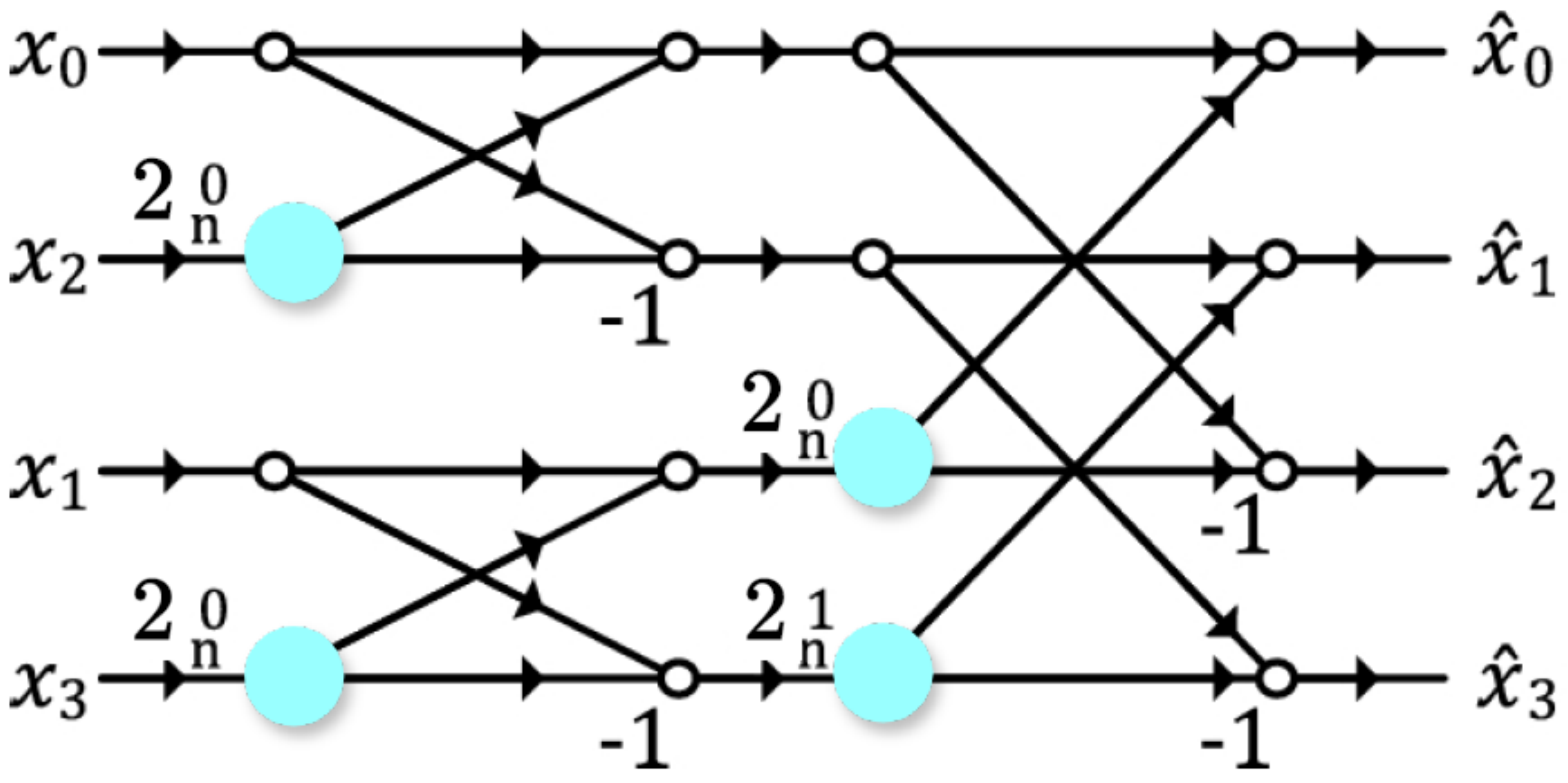
Approach: The Fermat Number Technique

- Fermat number, $P = 2^K + 1$ as auxiliary modulus

Advantages of the Fermat Number Technique

- Fermat number, $P = 2^K + 1$ as auxiliary modulus





NTT: Modular multiplications ⇒ FNTT: Simple shift operations

Advantages and challenges

- ☑ Multiplier-less NTT using Fermat number
- ☑ Roots of unity are powers of two $\implies$ no storage required
- ☑ We* achieve $1{,}200\,\times$ speed-up compared to software implementations
- ○ Requires more number of computations

Andrey Kim, Ahmet Can Mert, Anisha Mukherjee, Aikata Aikata, Maxim Deryabin, Sunmin Kwon, HyungChul Kang, and Sujoy Sinha Roy. **Exploring the advantages and challenges of Fermat NTT in FHE acceleration**. CRYPTO, 2024.

Outline

- **Background and Motivation**
  - Homomorphic Encryption (HE)
  - Ring-LWE based HE and challenges
- **FHE-HW: Hardware acceleration for HE**
  - FNTT: Fermat's Number Technique for NTT
  - REED: Chiplet-based hardware accelerator
- **HW-FHE: HE for hardware reusability**
  - ModHE: Module-LWE based HE scheme
- **Beyond HE**
  - Hybrid Homomorphic Encryption (HHE)
  - SASTA: Fault attack on HHE

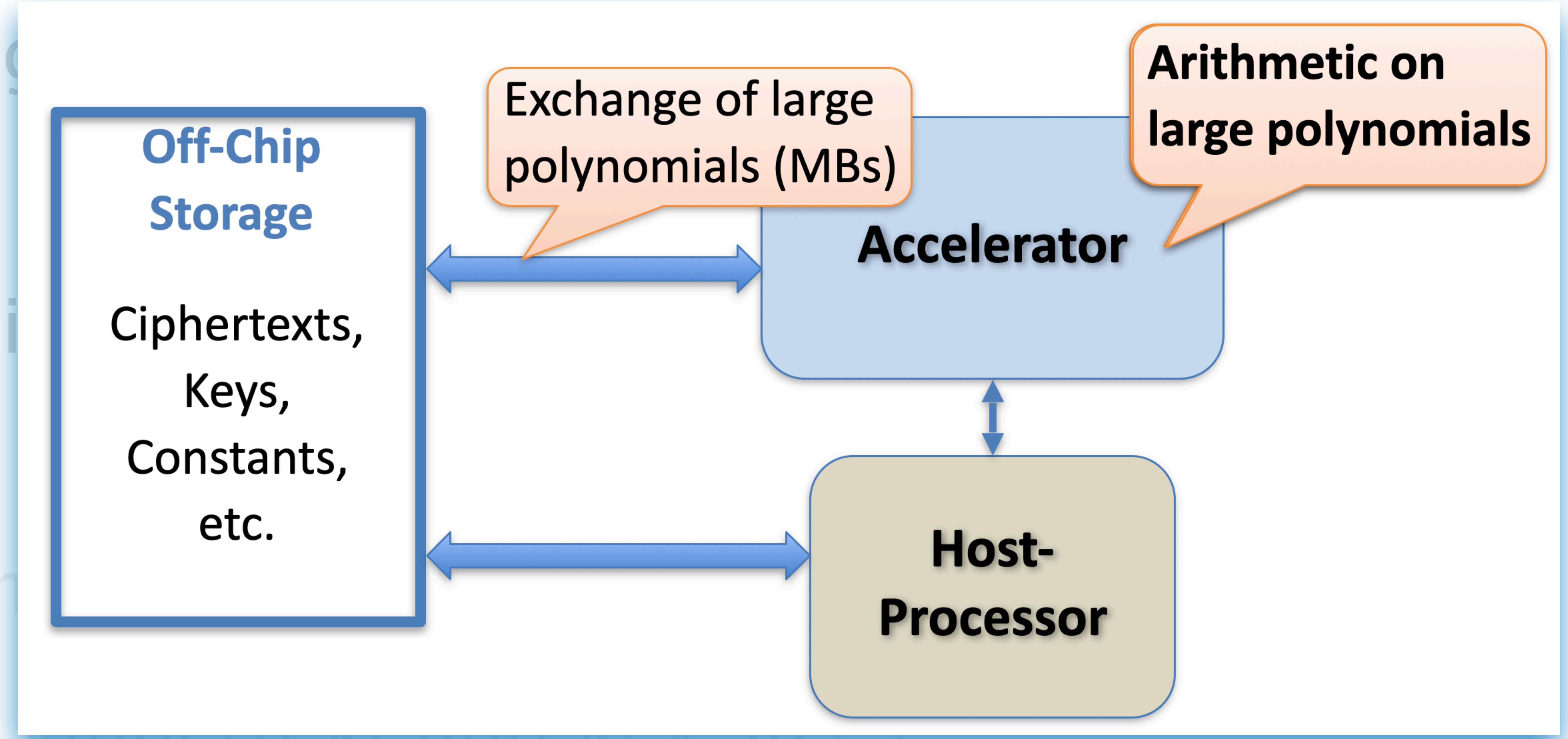
Challenges with FHE-HW acceleration

- Many (large) polynomial arithmetic operations
  - Large degree polynomial arithmetic
  - Long integer arithmetic

- Memory management
  - Large ciphertext and key sizes
  - Limited on-chip memory

Challenges with FHE-HW acceleration

- Many (lar
  - Large
  - Long i
- Memory r
  - Large
  - Limited on−chip memory



System−level view

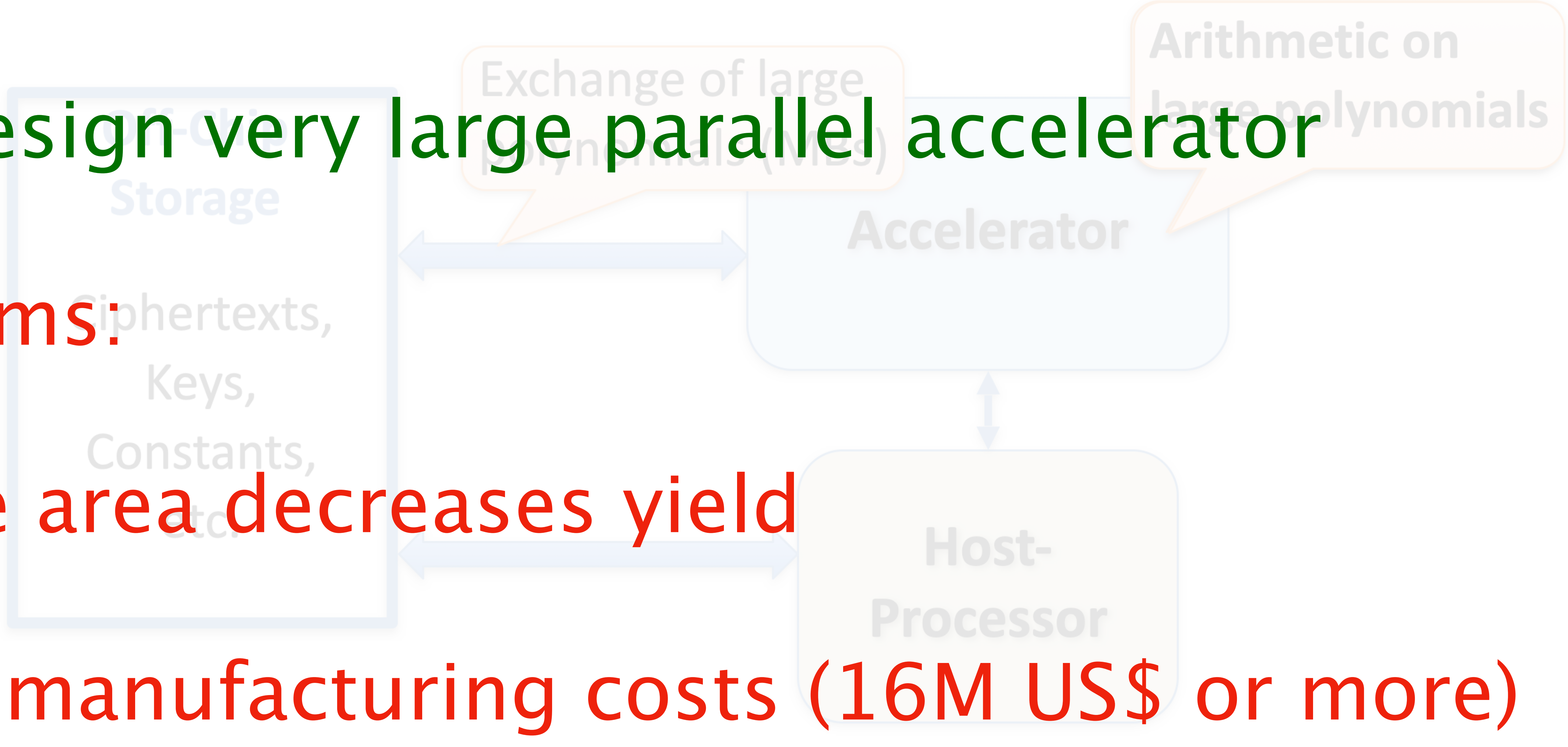
Challenges with FHE-HW acceleration

Problem: FHE is slow

Solution: Design very large parallel accelerator

New problems:

- Large area decreases yield
- High manufacturing costs (16M US$ or more)
- Difficult pre-silicon testing and verification



System-level view

Challenges with FHE-HW acceleration

Problem: FHE is slow

Solution: Design very large parallel accelerator

New problems:

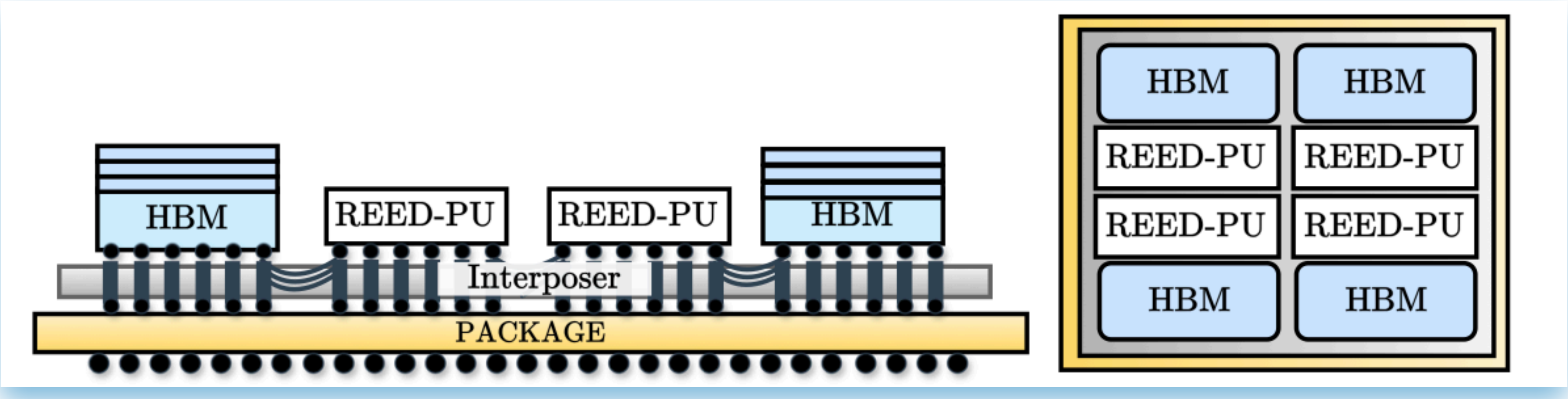
- Large area decreases yield
- High manufacturing costs
- Difficult pre-silicon testing and verification

**Picture credits: Tenor

Approach: Chiplet integration

- Split a big design into multiple dies called ‘chiplets’
  - Dies are 2.5D or 3D ‘packaged’

Chiplet-based FHE processor: REED*



Side and top view–2.5D REED

Aikata Aikata, Ahmet Can Mert, Sunmin Kwon, Maxim Deryabin, and Sujoy Sinha Roy. **REED: Chiplet-based accelerator for fully homomorphic encryption**. *https://eprint.iacr.org/2023/1190.*

Advantages and challenges of a chiplet-based design

- ☑ Higher yield
- ☑ Smaller and simpler chiplets
- ☑ Manufacturing feasibility
- ○ Slow chiplet-to-chiplet (C2C) communication
- ○ Optimal balance between area and number of chiplets is crucial

Overcoming the challenges

- ☑ Algrithmic tweaks to develop a ring-based FHE C2C protocol
- ☑ No performance penalty
- ☑ Linear interconnection complexity

ASIC designs such as REED's chiplet system could bring FHE calculations within 10x latency compared to plaintext calculations.

Outline

- **Background and Motivation**
  - Homomorphic Encryption (HE)
  - Ring-LWE based HE and challenges
- **FHE-HW: Hardware acceleration for HE**
  - FNTT: Fermat's Number Technique for NTT
  - REED: Chiplet-based hardware accelerator
- **HW-FHE: HE for hardware reusability**
  - ModHE: Module-LWE based HE scheme
- **Beyond HE**
  - Hybrid Homomorphic Encryption (HHE)
  - SASTA: Fault attack on HHE

Challenges with FHE-HW acceleration

- Many (large) polynomial arithmetic operations
    - Large degree polynomial arithmetic
    - Long integer arithmetic

More complex application →Larger (N, q)

Challenges with FHE-HW acceleration

- Many (large) polynomial arithmetic operations
  - Large degree polynomial arithmetic
  - Long integer arithmetic

- Software FHE is flexible but very slow
- Hardware is **fast but inflexible**

Can we design an HE scheme that allows the same hardware to support multiple (N, q)?

Approach: Module-LWE (MLWE) meets HE

Ring

$$a(x) \cdot s(x) + e(x) = b(x) \pmod q \pmod{f(x)}$$

$a(x) \in \mathcal{R}_q(N = 2^{14})$

$e_i \in \{0, 1, -1\}^N$

$s(x) \in \mathcal{R}_q(N = 2^{14})$

$b(x) \in \mathcal{R}_q(N = 2^{14})$

Module-LWE (MLWE) meets HE

Module

$$\begin{bmatrix} a_{00}(x) & a_{01}(x) \\ a_{10}(x) & a_{11}(x) \end{bmatrix} \cdot \begin{bmatrix} s_0(x) \\ s_1(x) \end{bmatrix} + \begin{bmatrix} e_0(x) \\ e_1(x) \end{bmatrix} = \begin{bmatrix} b_0(x) \\ b_1(x) \end{bmatrix} \pmod{q} \pmod{f(x)}$$

$a_{ij}(x) \in \mathscr{R}_q(2^{13})$

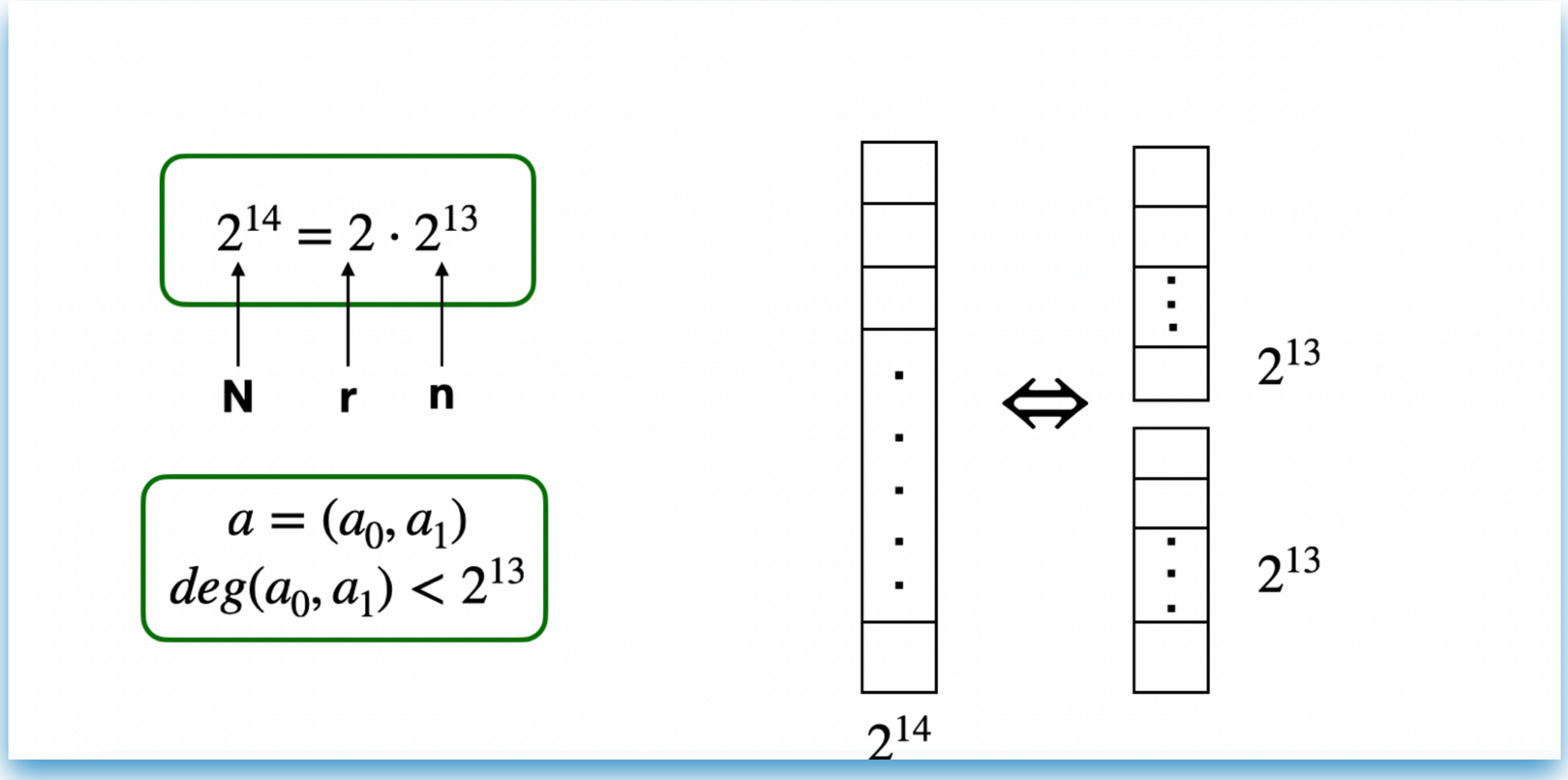
$s_i(x) \in \mathscr{R}_q(2^{13})$

$b_i(x) \in \mathscr{R}_q(2^{13})$

$$(N = 2^{14} = 2 \cdot 2^{13} = n \cdot r)$$

Module-LWE (MLWE) meets HE

- Flexible parameters: Fix a ring degree (n) and vary the rank (r)



Module-LWE (MLWE) meets HE: ModHE*

- Flexible parameters: Fix a ring degree (n) and vary the rank (r)

$$\Downarrow$$

Arithmetic on smaller and fixed degree (n) polynomials)

*Anisha Mukherjee, Aikata, Ahmet Can Mert, Yongwoo Lee, Sunmin Kwon, Maxim Deryabin, and Sujoy Sinha Roy. **Modhe: Modular homomorphic encryption using module lattices potentials and limitations**. *TCHES, 2024(1)*

ModHE: Potentials and limitations

- ☑ Better security assumptions
- ☑ Hardware reusability and more scope for optimization
- ☑ Increased scope of parallel computations
- ☑ Ciphertext compression due to rank reduction
- ○ Limitations: Increased key sizes, more precision loss

Outline

- **Background and Motivation**
  - Homomorphic Encryption (HE)
  - Ring-LWE based HE and challenges
- **FHE-HW: Hardware acceleration for HE**
  - FNTT: Fermat's Number Technique for NTT
  - REED: Chiplet-based hardware accelerator
- **HW-FHE: HE for hardware reusability**
  - ModHE: Module-LWE based HE scheme
- **Beyond HE**
  - Hybrid Homomorphic Encryption (HHE)
  - SASTA: Fault attack on HHE

Communication overhead in HE

m 🤝 pk 🔒

HE.Enc $\Longrightarrow$

$\texttt{ct} = (c_0, c_1) \in \mathcal{R}_q \times \mathcal{R}_q$

$f(\cdot)$ $\Big\Downarrow$ Eval(***m* + *m'*, *m* . *m'***)

$\texttt{ct}_{\text{add}} = \texttt{ct} + \texttt{ct}' \in \mathcal{R}_q \times \mathcal{R}_q$

$\texttt{ct}_{\text{mult}} = \texttt{ct} \times \texttt{ct}' \in \mathcal{R}_q^3$

HE.Dec $\Longleftarrow$

$< \texttt{ct}, s > = c_0 + c_1 \cdot s$

CLIENT | **SERVER**

Communication overhead in HE

m 🤝 pk 🔒 $\Longrightarrow$ HE.Enc $\texttt{ct} = (c_0, c_1) \in \mathscr{R}_q \times \mathscr{R}_q$

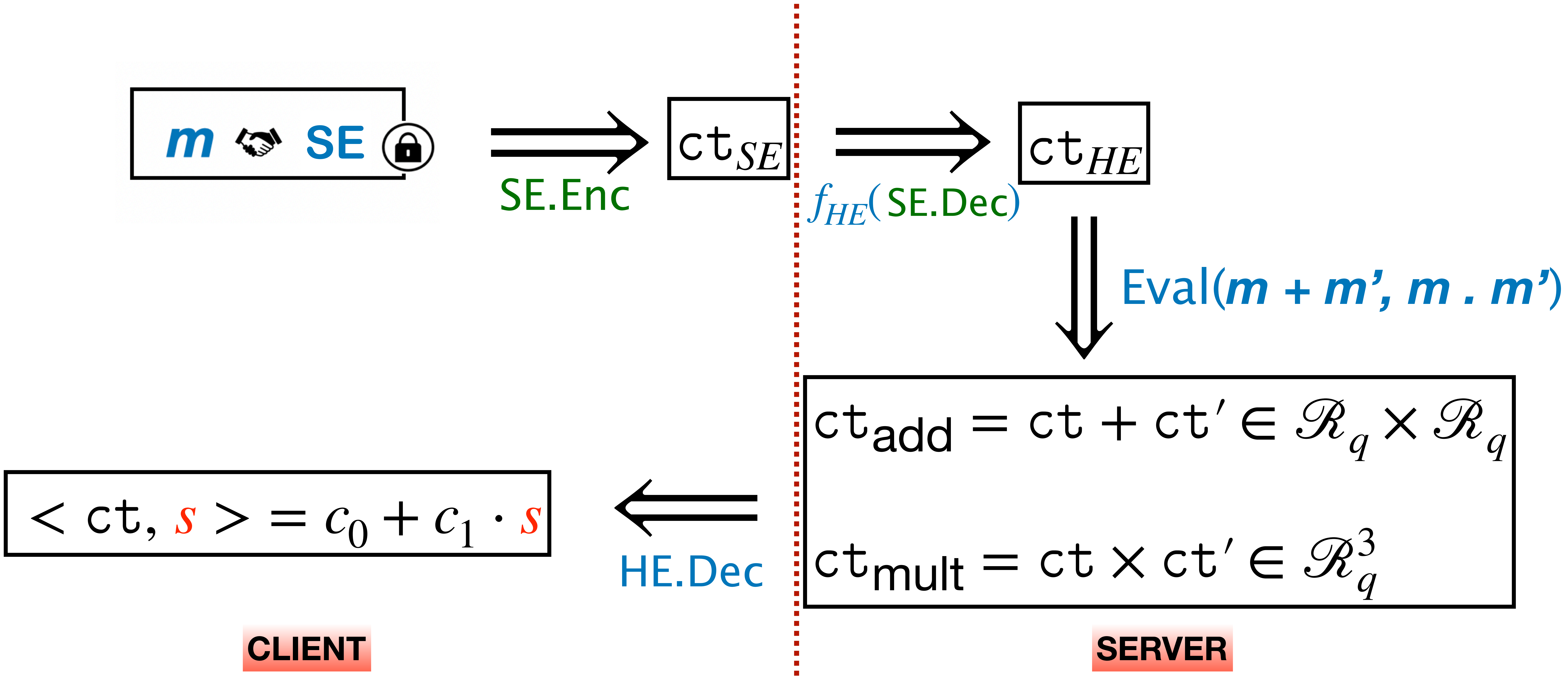
Huge ciphertext expansion
e.g., 7.4MB for ≤ 250kB

$\texttt{ct}_{\text{add}} \ldots \in \mathscr{R}_q \times \mathscr{R}_q$

$\texttt{ct}_{\text{mult}} = \texttt{ct} \times \texttt{ct}' \in \mathscr{R}_q^3$

$< \texttt{ct}, s > = c_0 + c_1 \cdot s$ $\Longleftarrow$ HE.Dec

Approach: Encryption using symmetric ciphers



Approach: Hybrid Homomorphic Encryption (HHE)

- Clients encrypts data symmetrically
  - No ciphertext expansion
- Server does more computations
  - Extra homomorphic decryption of symmetric circuit before eval

Approach: Hybrid Homomorphic Encryption (HHE)

- Clients encrypts data symmetrically
  - No ciphertext expansion
- Server does more computations
  - Extra homomorphic decryption of symmetric circuit before eval

Outline

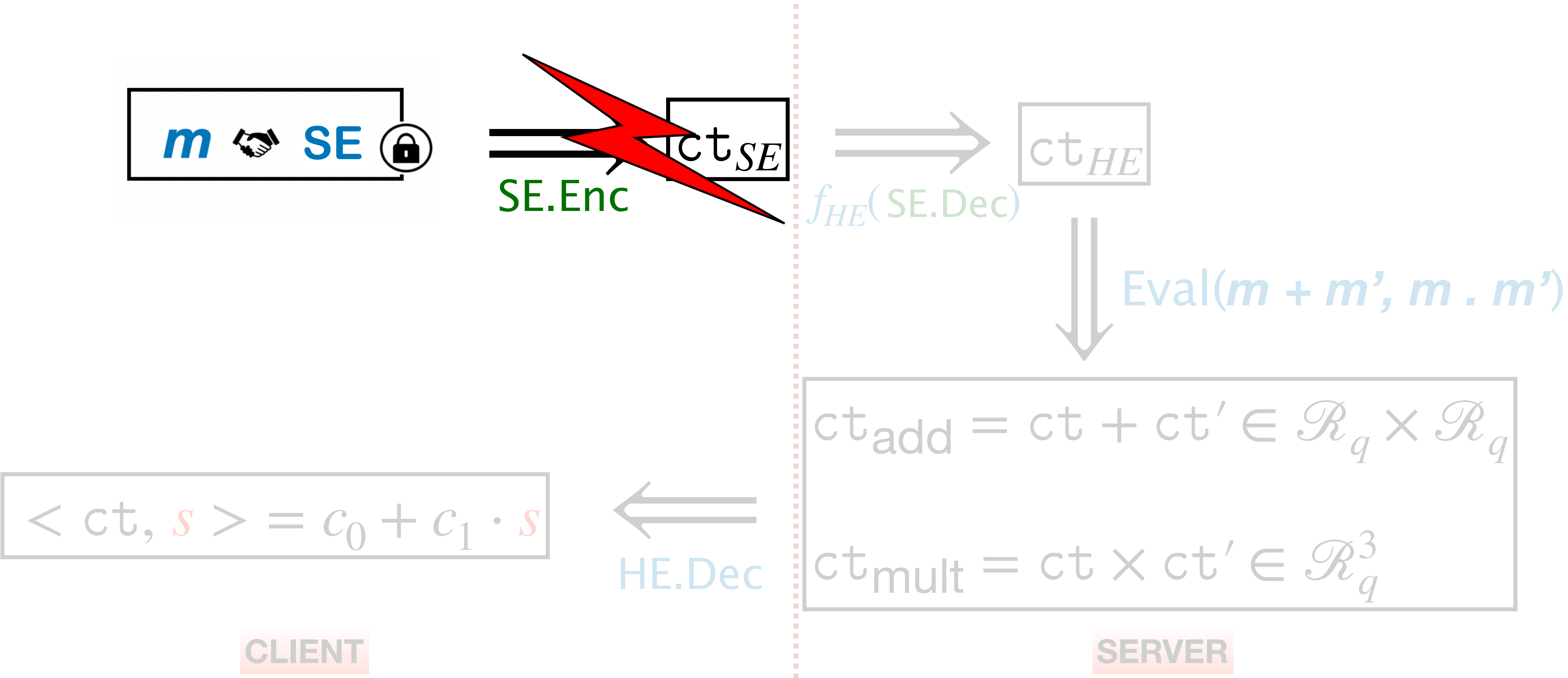
- **Background and Motivation**
  - Homomorphic Encryption (HE)
  - Ring-LWE based HE and challenges
- **FHE-HW: Hardware acceleration for HE**
  - FNTT: Fermat's Number Technique for NTT
  - REED: Chiplet-based hardware accelerator
- **HW-FHE: HE for hardware reusability**
  - ModHE: Module-LWE based HE scheme
- **Beyond HE**
  - Hybrid Homomorphic Encryption (HHE)
  - SASTA: Fault attack on HHE

HHE: Beyond theoretical security

- SASTA* introduces a novel fault attack on the SE.Enc step

Aikata Aikata, Ahaan Dabholkar, Dhiman Saha, and Sujoy Sinha Roy. **SASTA: Ambushing hybrid homomorphic encryption schemes with a single fault.** *https://eprint.iacr.org/2024/041.*

Beyond HE



**Picture credits: ClipArts

SASTA: Differential Fault Analysis

Differential = Faulty ciphertext – Faultfree ciphertext

$$= \texttt{ct}' - \texttt{ct}$$

$$= \left(m + \text{SE.Enc}(K_{SE}, n)\right) - \left(m + \text{SE'.Enc}(K_{SE}, n)\right)$$

$$\Delta E = \text{SE.Enc} - \text{SE'.Enc}$$

SASTA: Features and limitations

- ☑ Single fault at identified Fault Injection Points (FIPs)
- ☑ Single pair of faulty and fault-free ct required for key-recovery
- ☑ Demonstrated success for many HHE ciphers
- ○ Attack success dependent on complexity of evaluation function

Conclusion: Key-takeaways

- Homomorphic Encryption provides data privacy in untrusted environments
- Suffers from large computational overhead
- Interesting scopes in new hardware/computation paradigms & scheme design
- Interesting scopes for in-depth cryptanalysis

FHE for hardware, hardware for FHE and beyond!

Anisha Mukherjee

IAIK, Graz University of Technology, Austria

September 4, 2024