Designing Hardware for Cryptography and Cryptography for Hardware

Srini Devadas

Contributions from Nikola Samardzic, Axel Feldmann, Alex Krastev, Simon Langowski, Sacha Servan-Schreiber, Daniel Sanchez

Massachusetts Institute of Technology

MIT CSAIL

Outline

What can hardware do for security?

Accelerating cryptography

Should cryptography change with hardware accelerators?

What’s next?

Outline

What can hardware do for security?

Accelerating cryptography

Should cryptography change with hardware accelerators?

What's next?

Hardware Provides Secure Key Storage

Physical attacks

Invasive probing

Non-invasive measurement

Processor Architectures for Improved Security

- **Architect a processor to track the flow of information through the code**
  - This can be done in software albeit with greater overhead
  - CHERI, DOVER, **ARM Memory Tagging Extensions**
- **Architect a processor to provide strict isolation between applications minimizing trusted computing base (TCB)**
  - Secure compartments: XOM, AEGIS, Bastion
  - Enclaves: **Intel SGX**
  - Side channel resistance: Sanctum, STT, MI6, next version of Intel SGX (?)

What Else Can Hardware Do To Improve Security?

Can implement security functionality in hardware, e.g.,

- Encryption
- Message authentication
- Network packet inspection
- Etc.

to improve performance and lower energy

Advanced Encryption Standard (AES) Accelerators

- AES-NI instruction set – standard on modern CPUs.
- Accelerate common applications like TLS for secure web browsing.

- Many cryptographic applications use AES for better performance
  - PRGs, PRFs, Multi-party computation (VSS).
  - Faster software algorithms (compared to software AES) exist for these primitives, but hardware acceleration beats them.

Cryptography on the Rise!

- Private Information Retrieval
- Oblivious Random Access M)
- Garbled Circuits
- Multiparty (MPC)
- Fu ic Encryption (FHE)
- e Computation (VC)
- ero-Knowledge Proofs, e.g., zk-snarks

Computationally Intensive!

Hardware Acceleration Can Enable NEW Use Cases

Outline

What can hardware do for security?

Accelerating cryptography

- Case study: Fully Homomorphic Encryption (FHE)

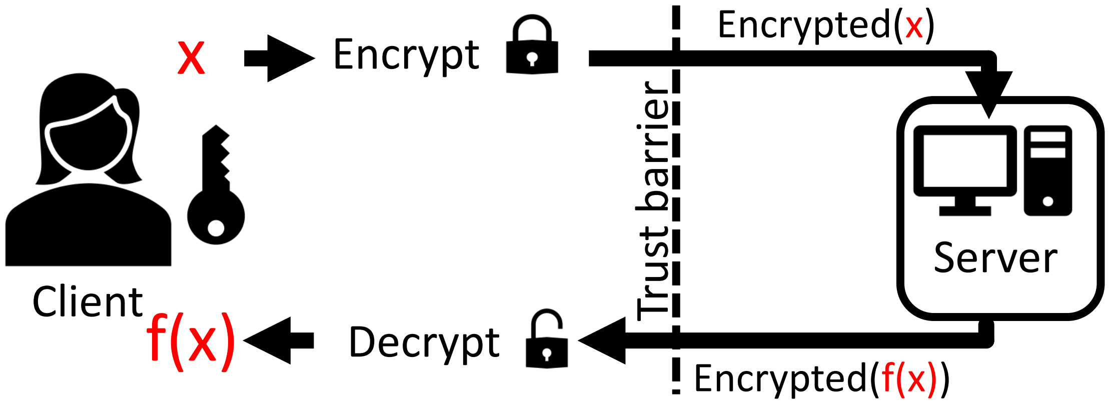
Should cryptography change with hardware accelerators?

What's next?

Fully Homomorphic Encryption (FHE)

The Holy Grail of secure computing

- FHE is a family of encryption schemes that compute directly on encrypted data
- FHE enables secure offloading of computation to the cloud:

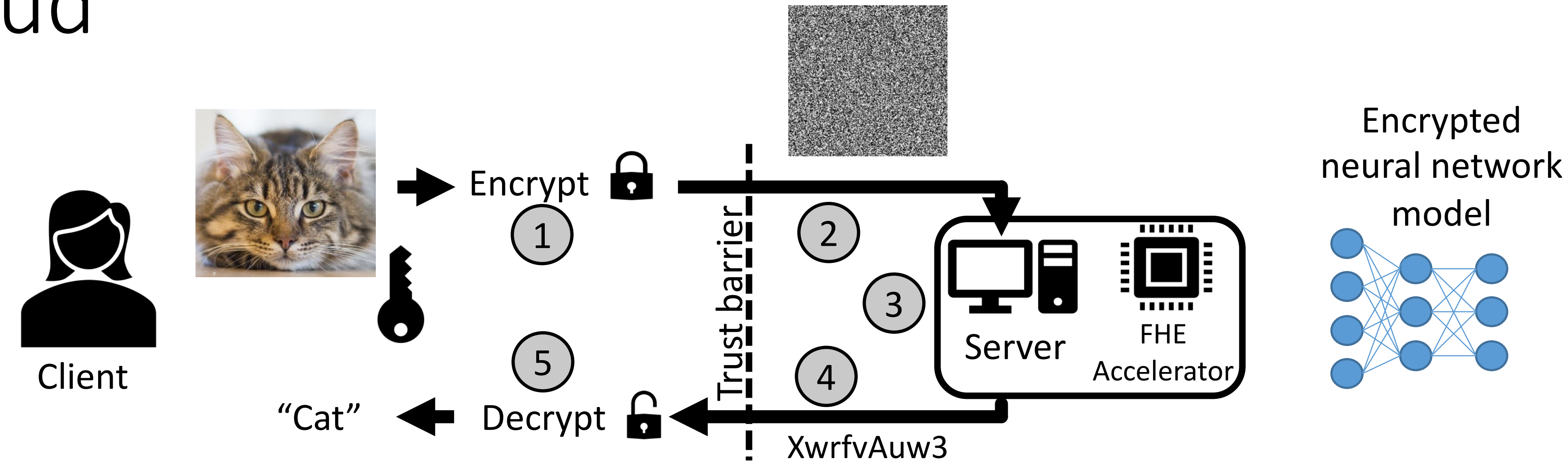


- Perfect security, since server never decrypts data!

Interest in FHE is exploding

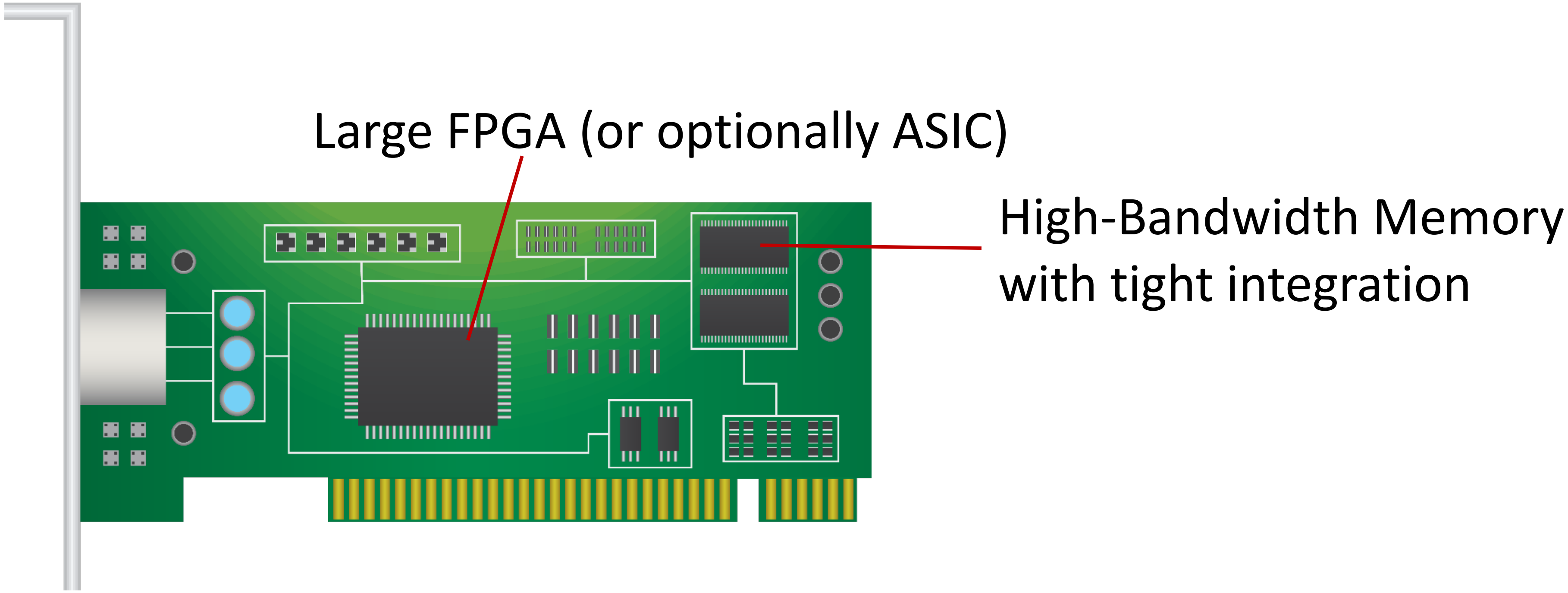
- 2009: First FHE scheme realized [Gentry]
  - 10^9 times slower than unencrypted computation
- 2012+: Improved FHE schemes [BGV, B/FV, CKKS, …]
  - Richer operations, reduced overheads
  - 10^4 to 10^6 times slower than unencrypted computation
- 2019: First FHE system that allows deep learning inference in under 1h [LoLa]
- 2021: Intel, IBM, Google, and others announce major FHE initiatives

Example Application: Private Inference in the Cloud



- Client can perform inference in the cloud without revealing its inputs
- Neural network model (weights) can be encrypted and hidden from server or in plaintext
- Use cases: Model must remain private to client or too large to download to client
- **State of the art without acceleration: 20 minutes per DNN inference**

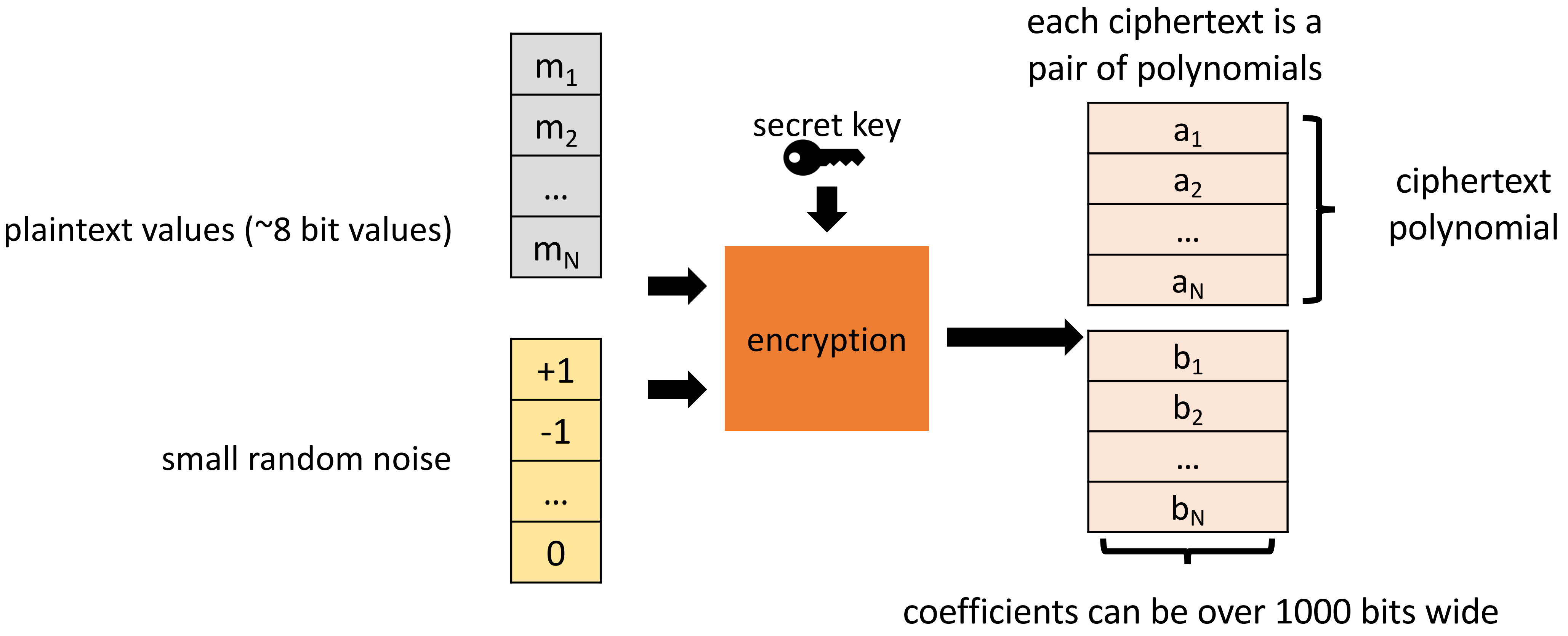
Secure computing accelerator



- Private deep learning on small to medium-sized models
  - ~1s/inference (FPGA) or ~100ms/inference (ASIC)
- Scaling with multiple chips/board

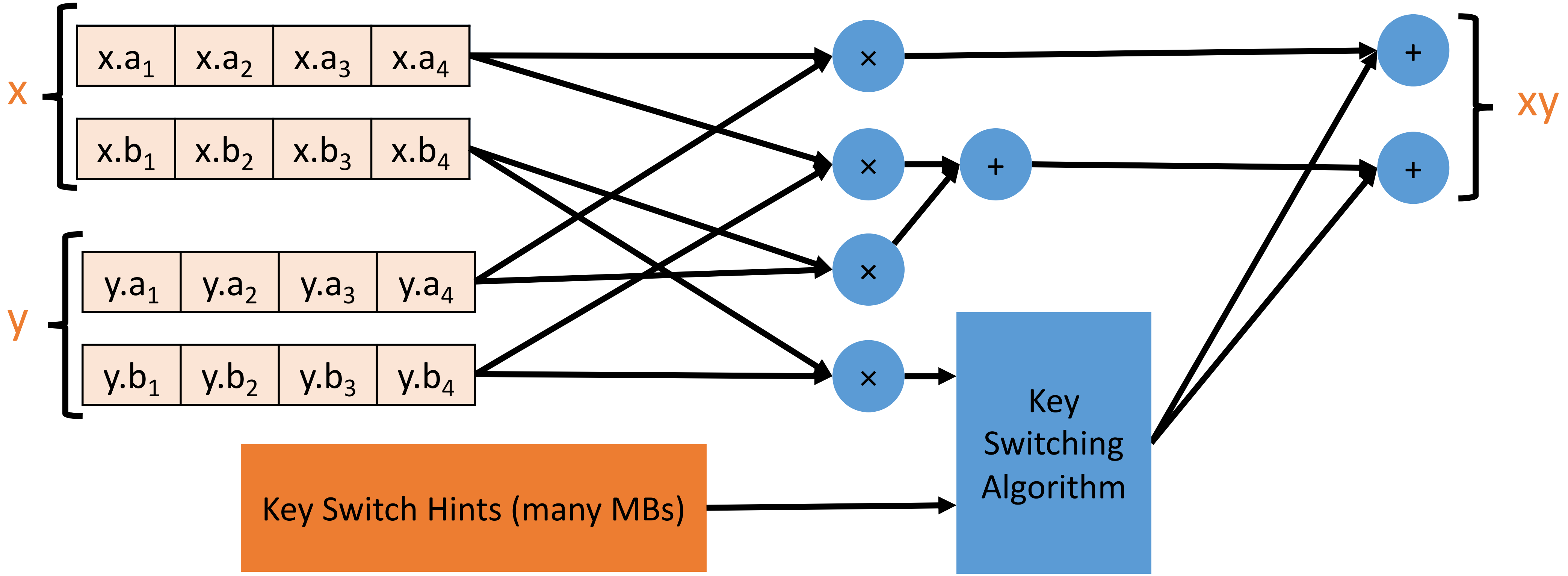
Encryption and Data Types

- Most FHE schemes encrypt vectors of numbers
- Plaintext vectors are encrypted into pairs of **polynomials**
  - Polynomials are represented as vectors of coefficients



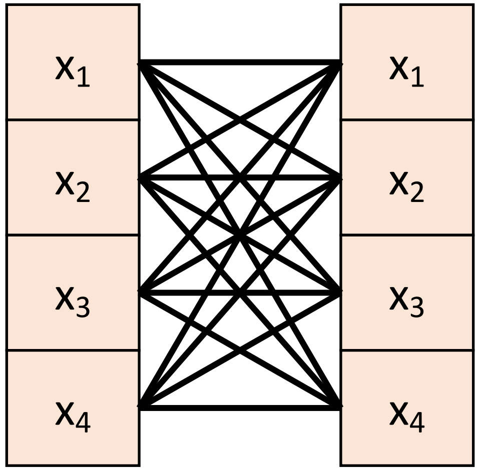
FHE Operations

- By computing on the ciphertext polynomials, FHE allows us to **add**, **multiply**, and **rotate** the underlying values
  - Operations on ciphertexts are often quite complex
  - Example: to multiply two ciphertexts **x** and **y**:



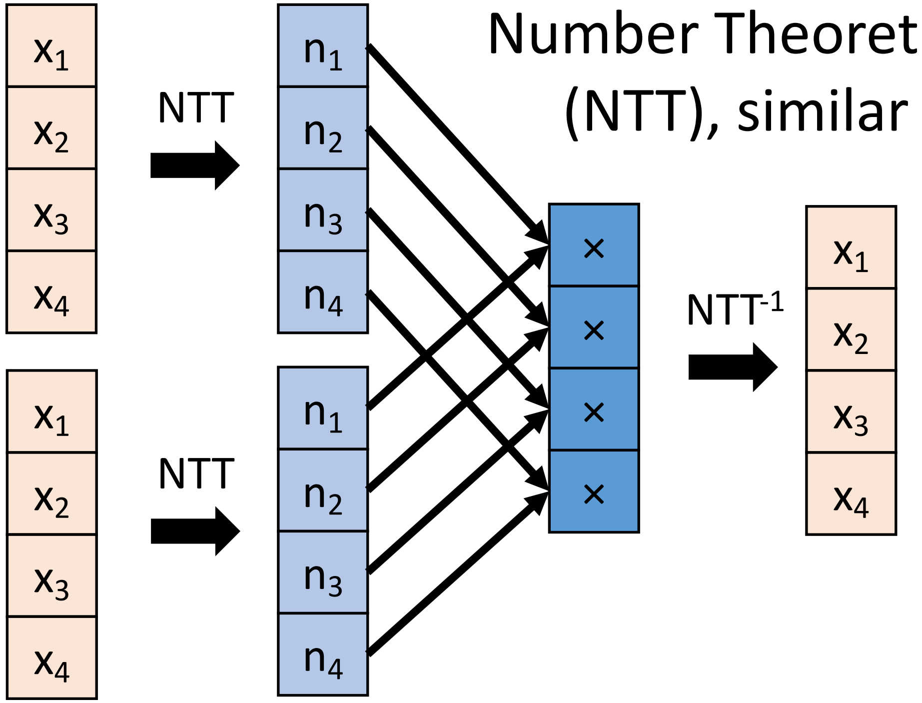
Multiplying Polynomials

- We often need to multiply polynomials



Naively, this takes $O(n^2)$ multiplications

Instead, we can use the Number Theoretic Transform (NTT), similar to an FFT



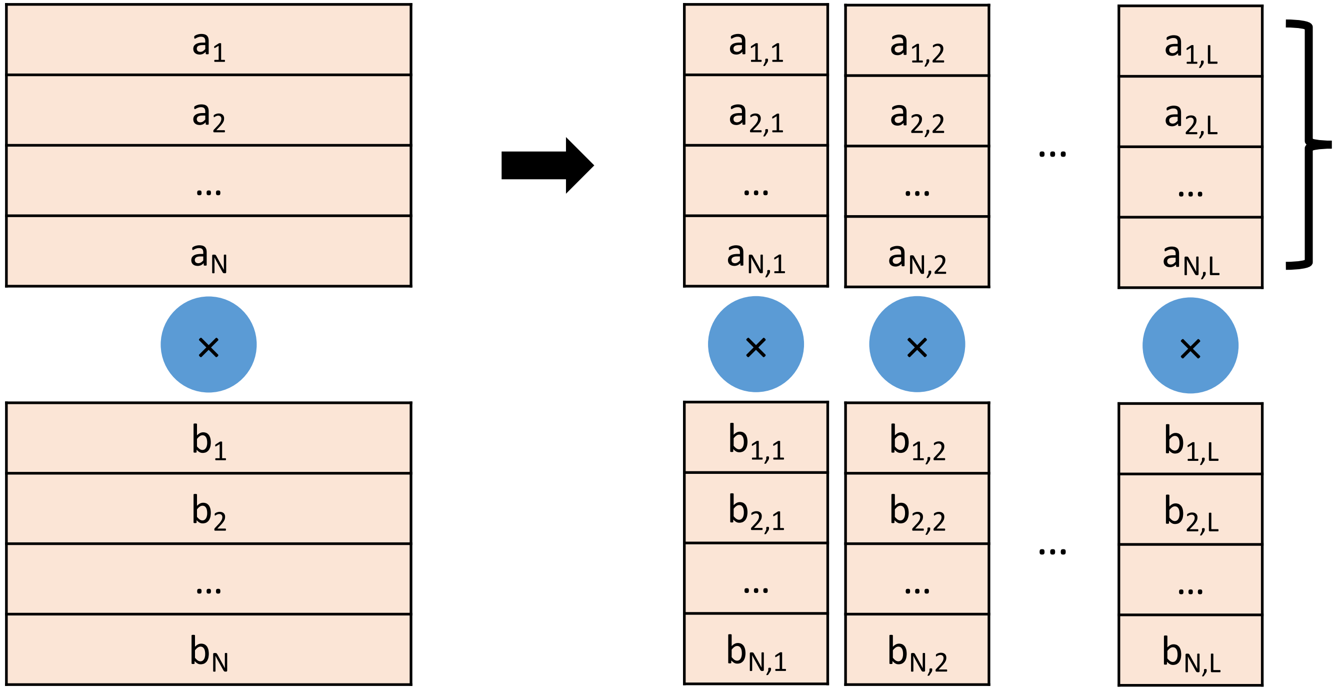
NTTs and NTT^{-1} each take O(nlogn) multiplies, making the whole operation O(nlogn)

Wide Arithmetic Using RNS

- Polynomial coefficients are extremely wide (over 1000 bits)
  - We also need to support computation on narrower ones

Residue Number System: we can represent a single wide polynomial modulo some large Q as L many polynomials each mod a smaller q_i where $q_1 q_2 \ldots q_L = Q$

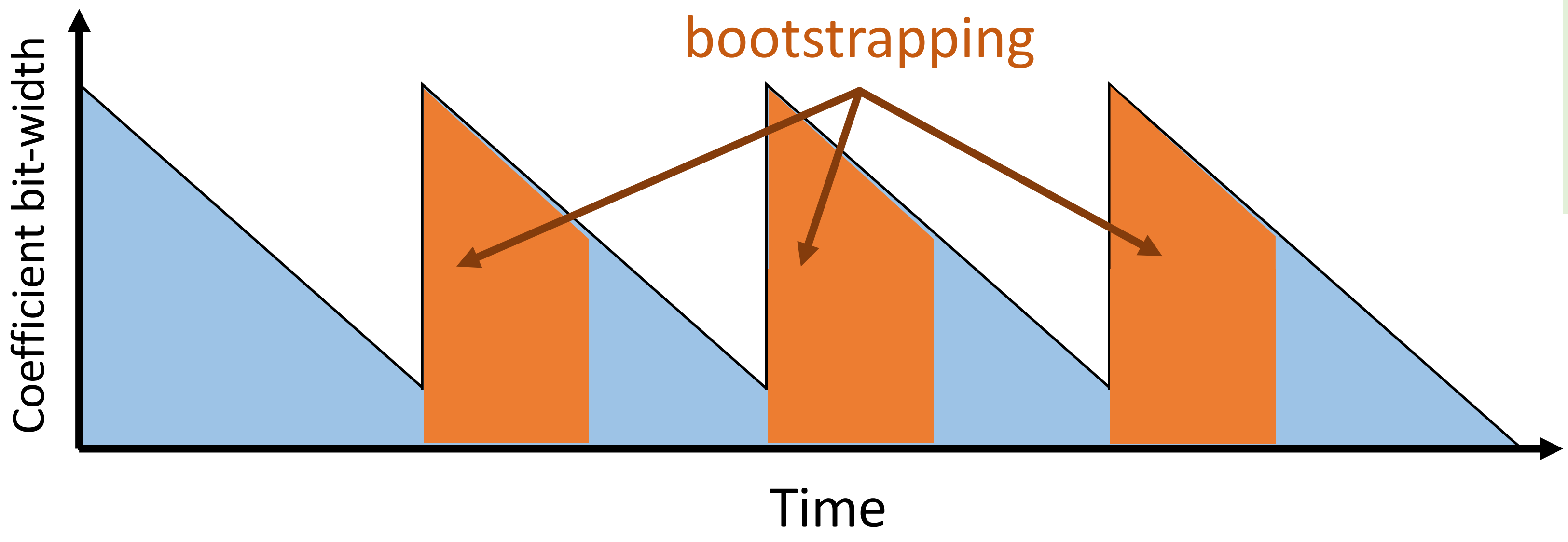
Advantage: we can perform arbitrarily wide modular arithmetic with narrow (~32-bit) multipliers



Residue polynomials are F1's primitive datatype

Rough Shape of FHE Programs

- Ciphertexts start with some initial noise and coefficient width
- As we compute on them, they become noisier, and we chop off the noise, also reducing the coefficient width
- **Bootstrapping** is an expensive procedure to refresh ciphertexts



Must perform computation at multiple bit-widths!

Architectural Characteristics of FHE

- FHE enables many algorithms on encrypted data, not just a single application
- Homomorphic operations all rely on big-polynomial arithmetic
- Ciphertexts are large (several megabytes), so data movement is extremely important
- **Dataflow is completely static**

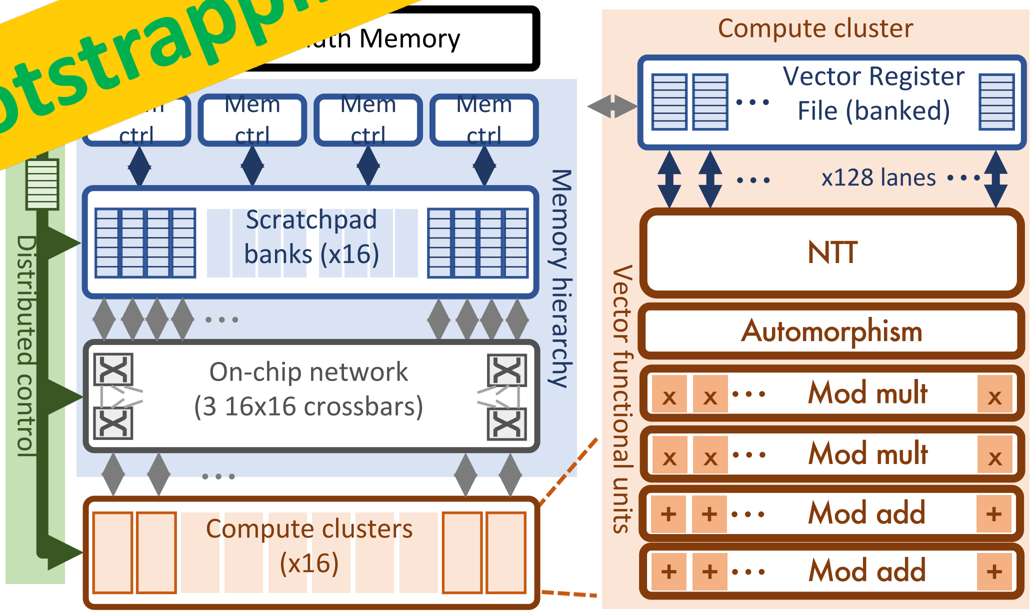
F1 Processor

MICRO 2021

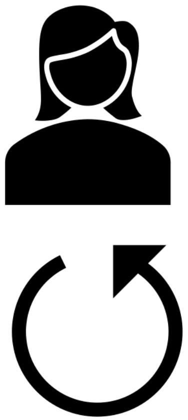
- Existing accelerators like GPUs ineffective on FHE because of data movement → Need new hardware and systems

- F1, an architecture specialized for FHE
  - First programmable FHE accelerator … executes full FHE programs
  - Hardware tailored to F… Modular arithmetic, novel f… …its, designed to minimize d…
  - Synthesis-b… evaluation on GF 12nm: 150mm^2 chip, 60-90 watts
  - **3000x-12,000x faster than CPU on deep learning; 200x-1,900x faster than GPUs, prior FHE accelerators**

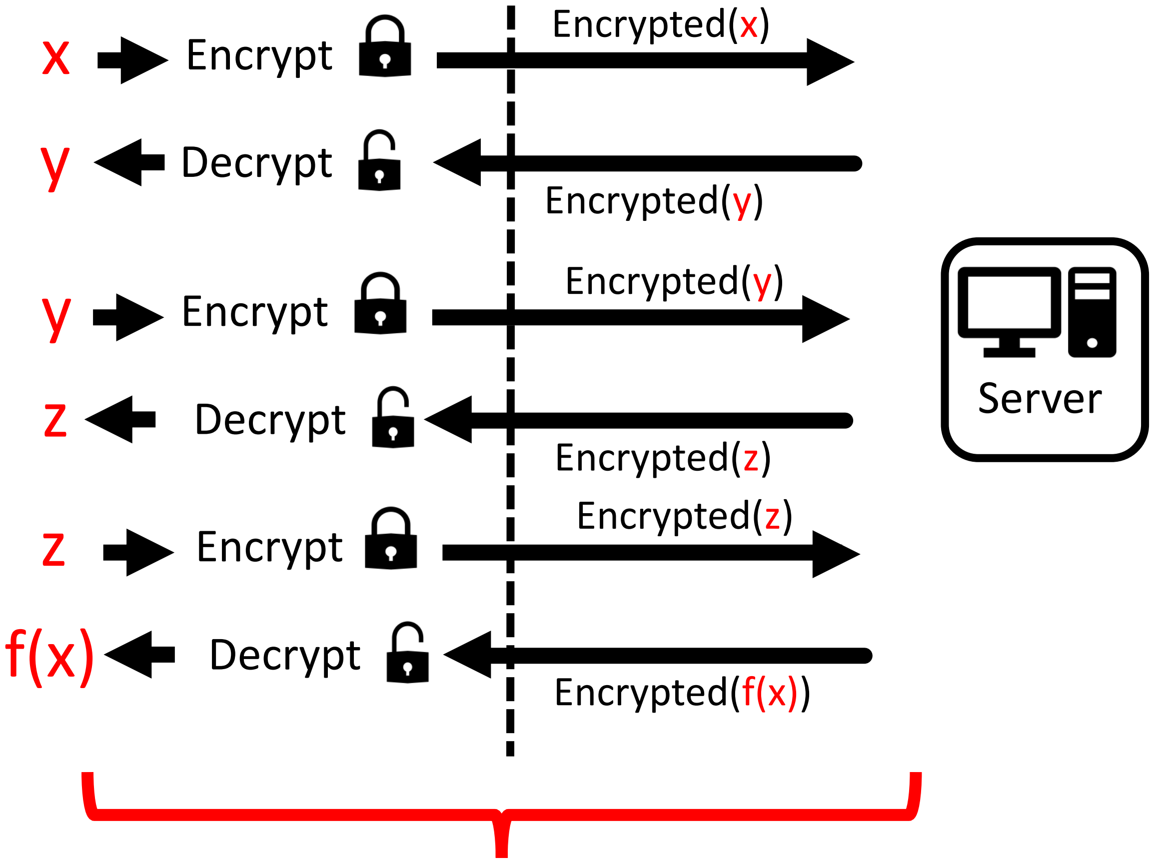
Does Not Accelerate Bootstrapping



Levelled FHE

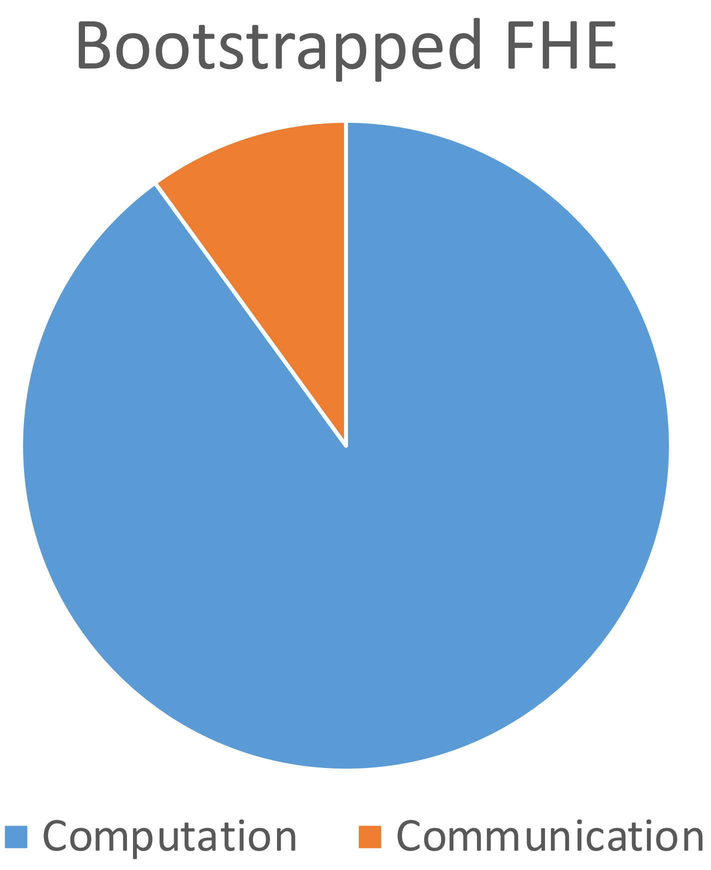


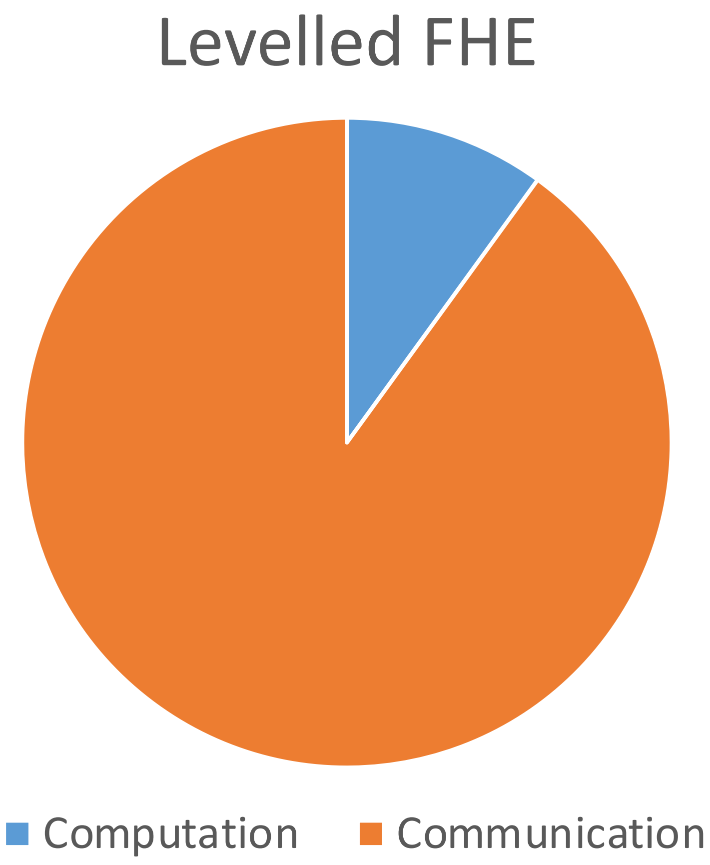
Re-encrypt intermediate values



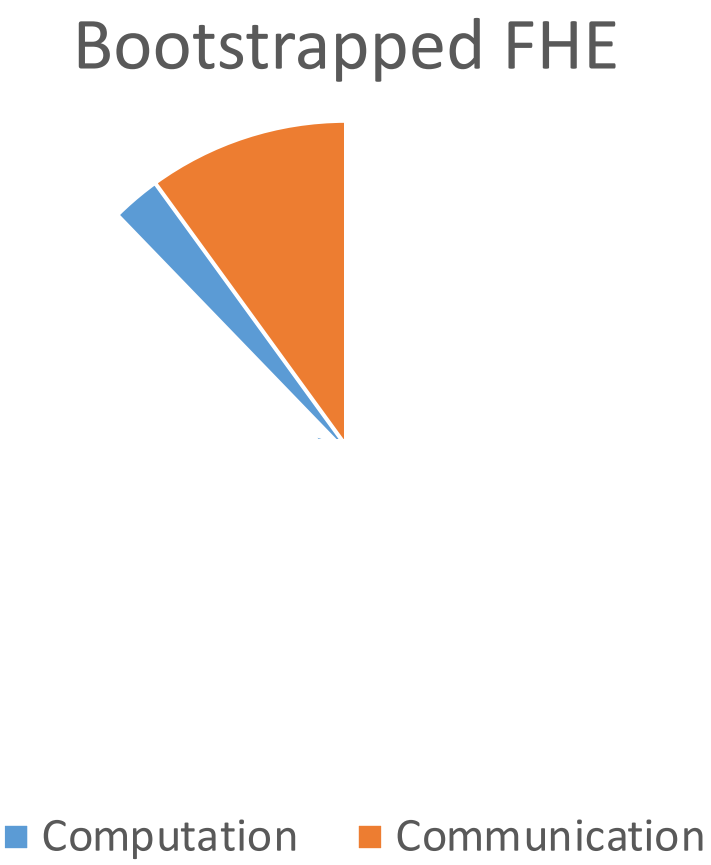
Large communication cost!

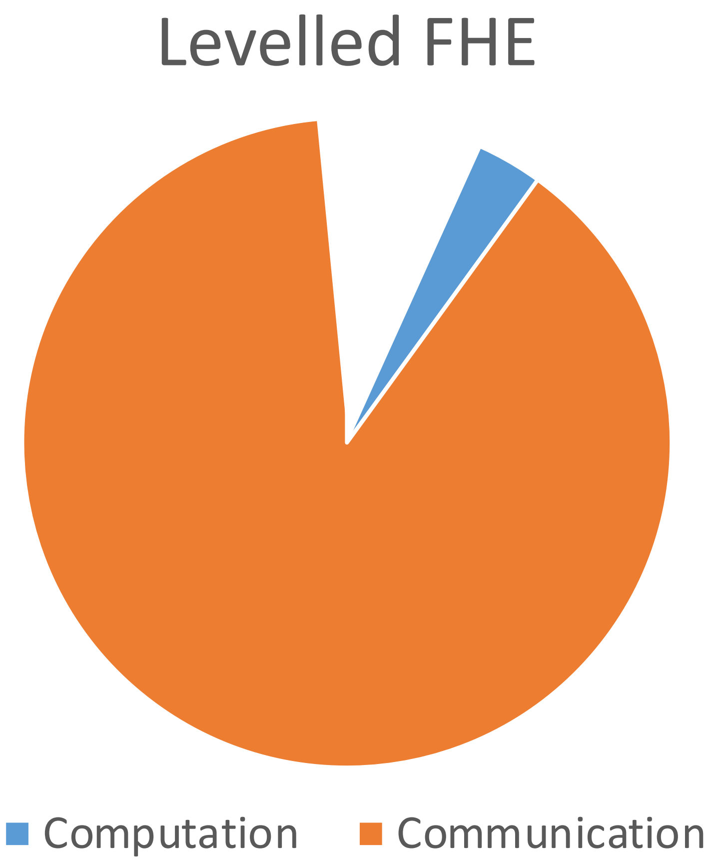
Bootstrap vs Levelled FHE





Bootstrap vs Levelled FHE *with acceleration*





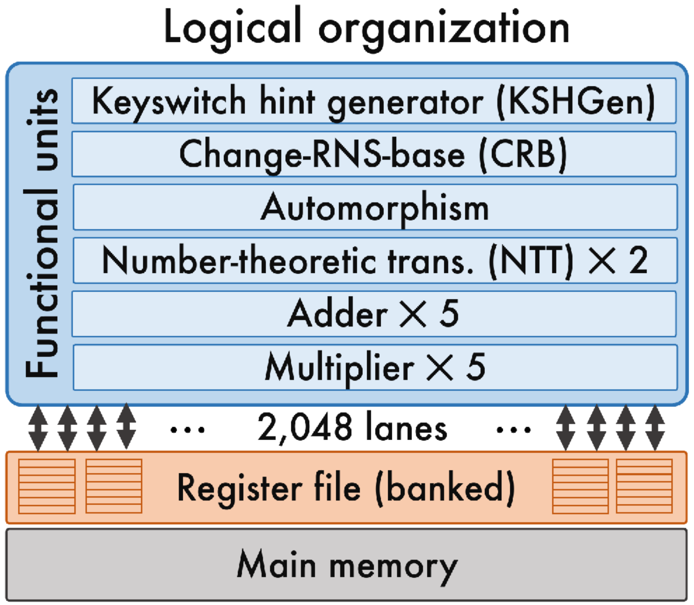
Observations

- Accelerating communication: **hard** (or in many cases, already done with hardware)
  - Network interface cards
  - Hardware for switches and routers
  - Physical layer

- Accelerating computation: **Need clever and careful design that speeds up ALL computation → Amdahl's Law**

CraterLake Architecture Overview

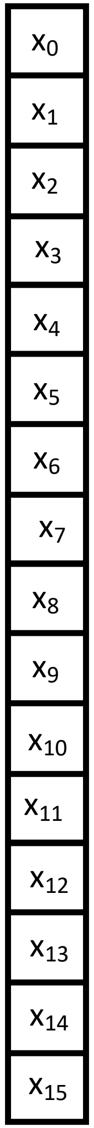
- Logical organization: A single set of *extremely wide* vector FUs (2048 lanes)
  - No compute clusters, unlike F1
- All on-chip storage in a large *single-level register file* (256 MB)
  - No RFs + scratchpad, unlike F1
- New FUs to accelerate boosted key switching (CRB, KSHGen)
- FUs can be chained to form *high-throughput pipelines*
- Static control, like F1

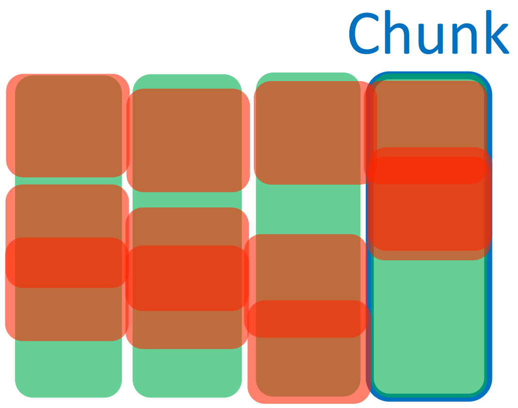


- Physical organization: Spatially distributed lane groups with an extremely simple fixed permutation network
  - Relies on new way to tile computation

CraterLake Vector Datapath

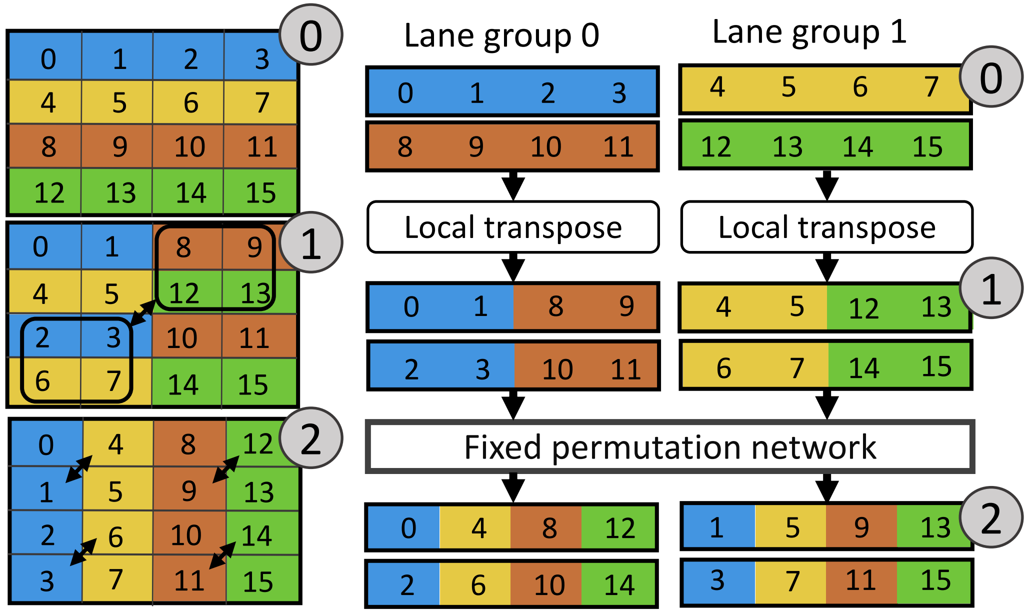
- Polynomials divided into E=2048-element **chunks**
- Datapath is 2048 lanes wide
- Vector adds and multiplies act coefficient-wise → easy to pipeline
- NTTs and automorphisms have dependencies *across chunks*, making them hard to vectorize and pipeline





Scaling the Vector Datapath with Transposes

- Key insight: NTTs and automorphisms can be decomposed by using transposes (one per NTT, two per automorphism)
- CraterLake tiles polynomial coefficients across lane groups
  - Each polynomial spans the whole chip, unlike in F1, where each polynomial was processed in a separate compute cluster
  - Improves throughput and reduces footprint, enabling much larger polynomials
  - With boosted key switching, this tiling incurs less global traffic than F1
- New 2-level transpose operation enables the global interconnect to be a *simple fixed permutation network*



Synthesis results

- Targeting ASIC evaluation
- Synthesized RTL on GF 12nm
- 1 TB/s HBM main memory (like F1)
- Similar budget to GPUs, server processors

- Compared architectures:
  - Server CPU baseline (32-core AMD Threadripper, similar area in 7nm)
  - F1+: Scaled-up F1 to have same compute capabilities (35% more area)

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**
Register file (256 MB)	192.0
On-chip interconnect	10.0
Mem. PHYs (2×HBM2E)	29.8
Total CraterLake	**472.3**

CraterLake performance

ISCA 2022

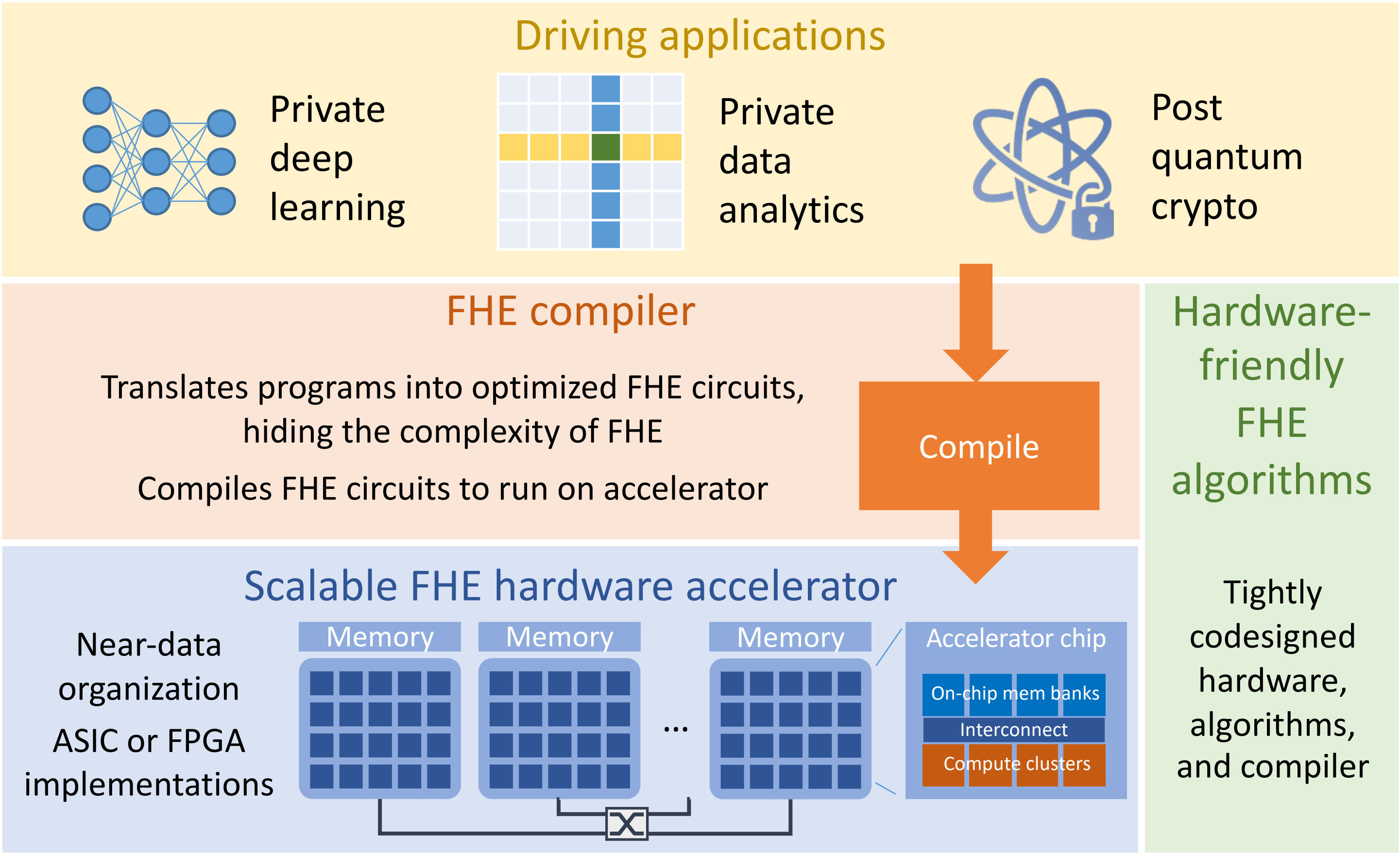
32-core CPU

Execution time (ms) on	CraterLake	F1+	CPU	vs. F1+	vs. CPU
ResNet-20	264.25	2,693	23 min	10.2×	5,211×
Logistic Regression	120.74	639	[illegible]	5.29×	2,949×
LSTM	128.90	2,573	[illegible]	[illegible]0.0×	6,665×
Packed Bootstrapping	2.76	[illegible]	[illegible]	21.1×	6,228×
deep gmean speedup				**12.3×**	**5,025×**
Unpacked bootstrappi[illegible]	[illegible]	0.21	877	2.04×	8,612×
CIFAR Unencr[illegible]	58.22	94.1	187 s	1.62×	3,205×
MNIST U[illegible]	0.15	0.13	561	0.88×	3,771×
MNIST Enc[illegible]ghts.	0.26	0.22	1369	0.84×	5,297×
shallow gmean speedup				**1.25×**	**4,846×**

$100M to Build and Deploy!

- CraterLake outperforms F1+ by up to 21x on deep benchmarks; similar performance on shallow benchmarks

A full-stack approach from hardware to algorithms



Outline

What can hardware do for security?

Accelerating cryptography

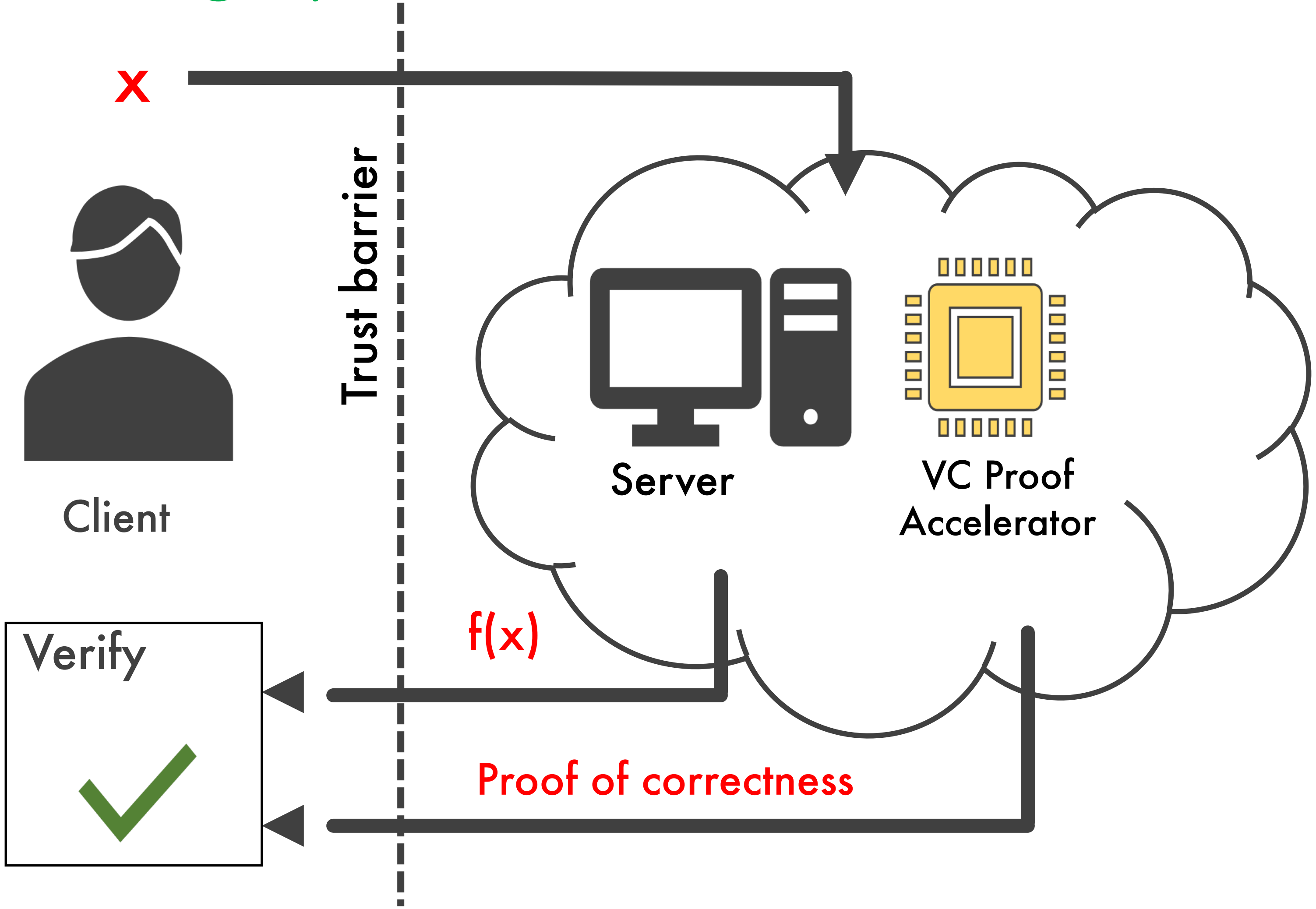
- Case study: Fully Homomorphic Encryption (FHE)

Should cryptography change with hardware accelerators?

What’s next?

Verifiable Computation (VC)

From Privacy to Integrity



zk-SNARK used in cryptocurrencies is a verifiable computation primitive

VC and FHE Computations Have Similarities

- Both FHE schemes and VC schemes rely on arithmetic over high-degree polynomials with large coefficients
- Both domains can share the same compiler front-end, which compiles a high-level language program into an arithmetic cryptographic circuit
- The dataflow graph of FHE programs and VC proof generation is known at compile time
- Both domains have operands that are up to tens of megabytes in size.
    - Must manage memory bandwidth effectively, while only being able to keep fewer than 10 operands on-chip

VC and FHE Computations Have Differences

- Elliptic-curve VC uses prime moduli that are hundreds of bits wide
- While FHE has even wider composite moduli, they can be decomposed into many independent smaller leveraging RNS
  - F1 and CraterLake accelerate FHE using relatively narrow word sizes (~30 bits wide)
- Elliptic-curve VC requires hardware that supports very wide (100s of bits) arithmetic

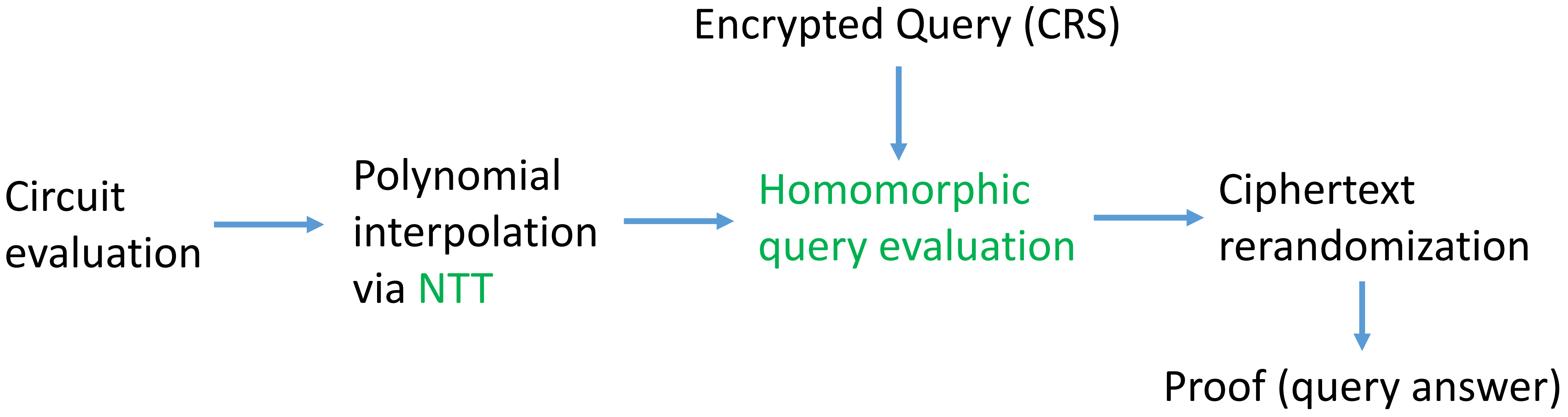
Using CraterLake for VC

- Instead of using an elliptic-curve based VC scheme, we can use a lattice-based one.

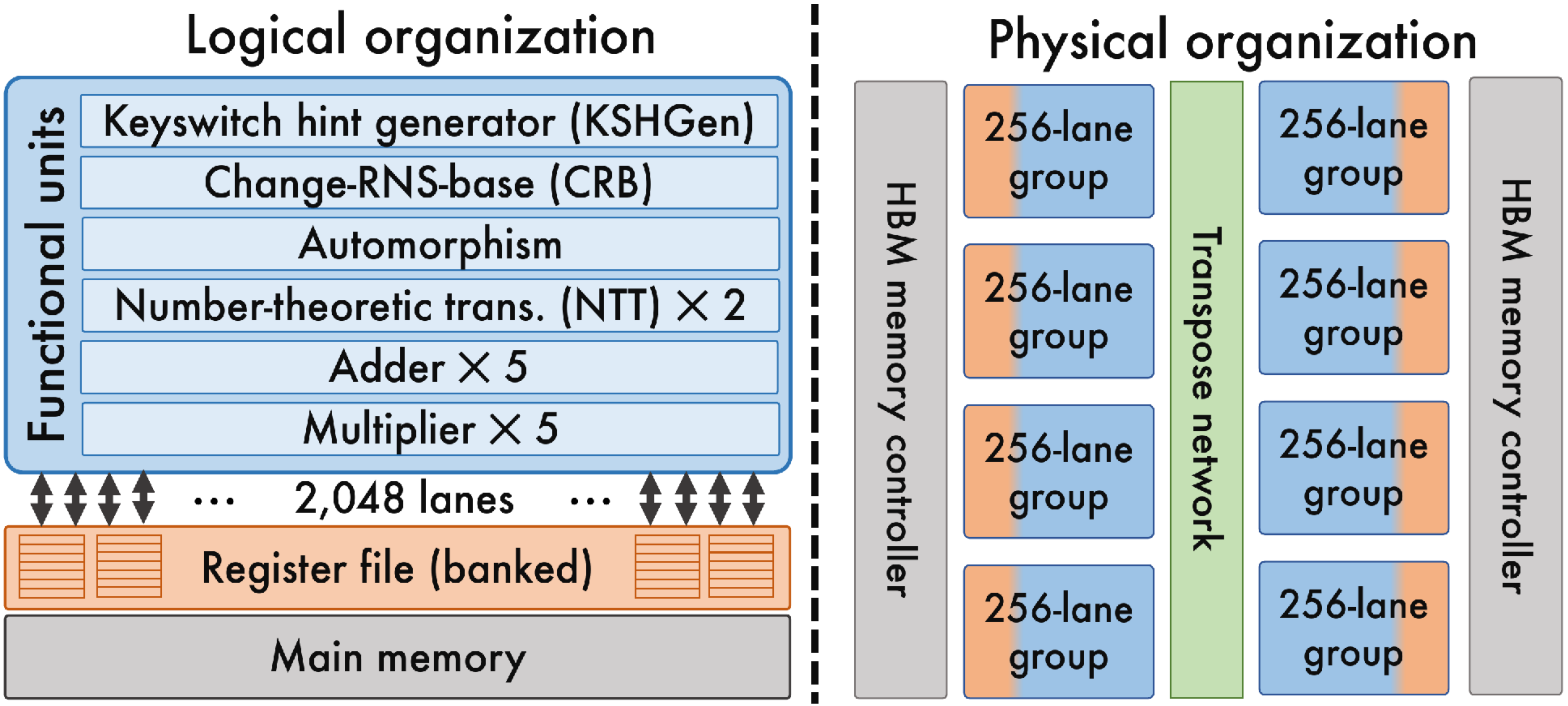
- Example: "Shorter and Faster Post-Quantum *Designated-Verifier* zkSNARKs from Lattices."
- Combines the Peikert-Vaikuntanathan-Waters (PVW) lattice-based encryption scheme with the Groth-16 zkSNARK proof system.
  - Doesn't require wide arithmetic in large prime fields
  - Not public-key verifiable

- Can also use CraterLake to accelerate other lattice encryption schemes!

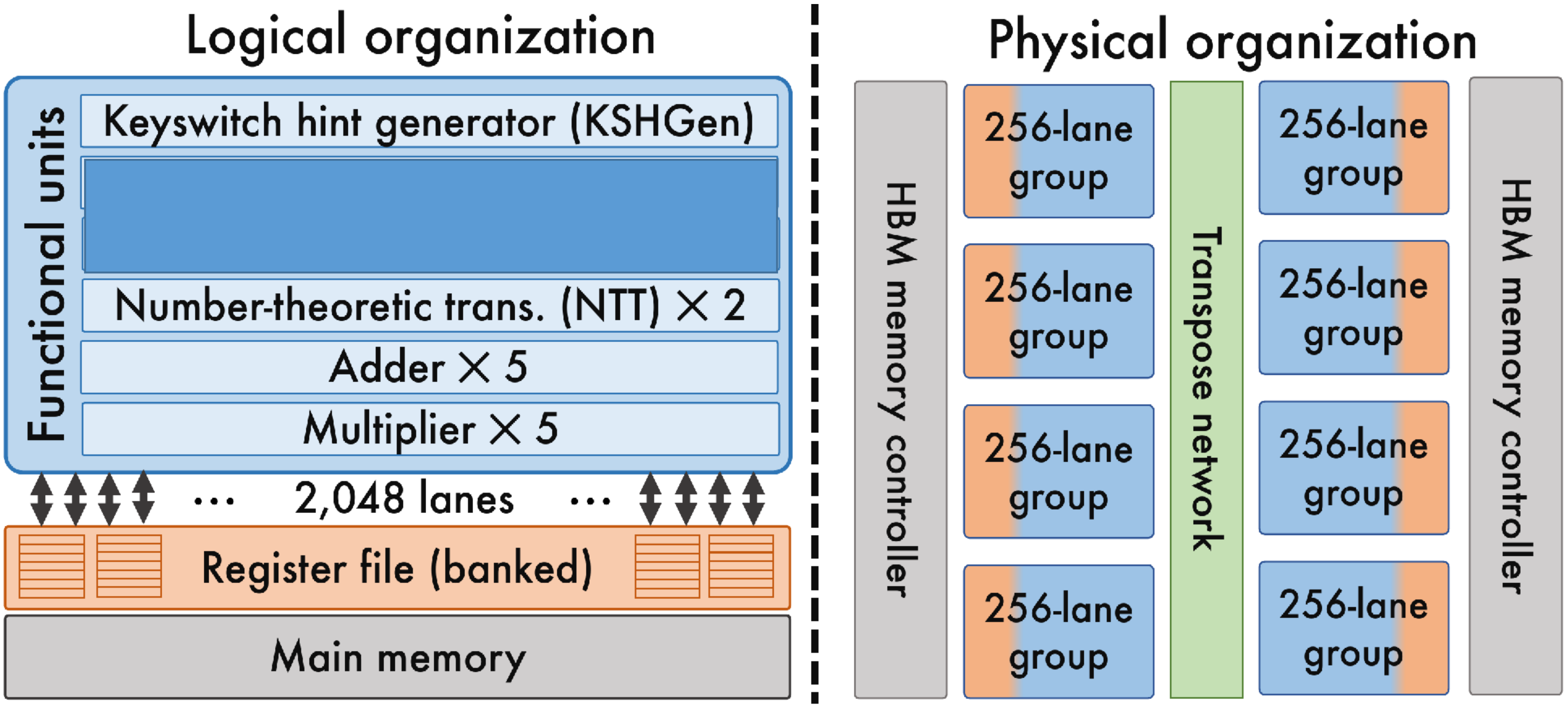
Structure of a Lattice-Based VC scheme



CraterLake Architecture (Revisited)

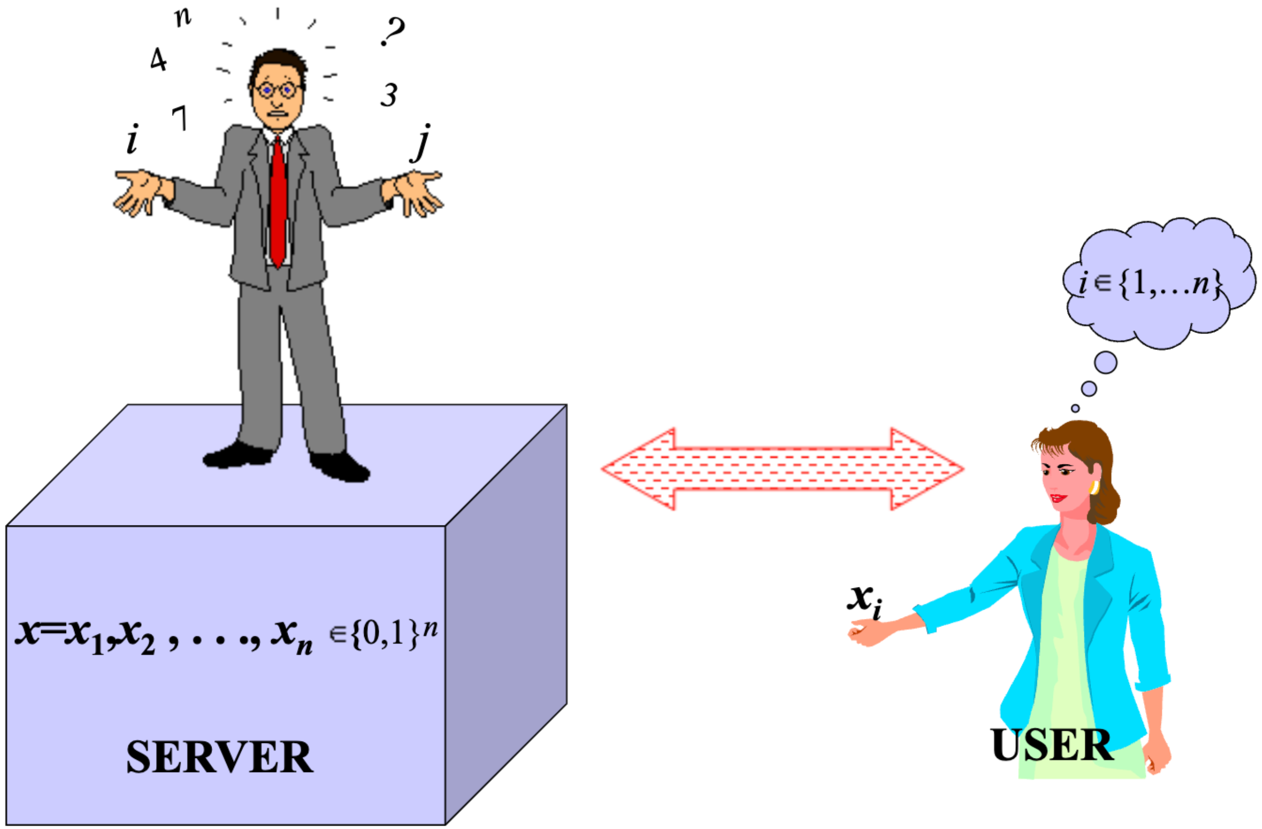


VC-only CraterLake



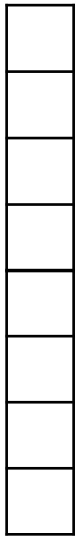
Private Information Retrieval (PIR)

- Allows a user to retrieve an item from a server in possession of a database without revealing which item is retrieved

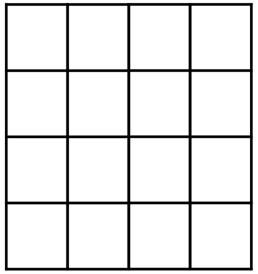


Courtesy: Beimel, Ishai, Kushilevitz, Malkin

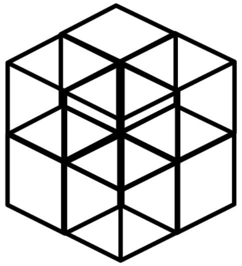
PIR Recursion Level



1-D database



2-D database



3-D database

Communication: **O(n)**
send the whole database
Computation: *None*

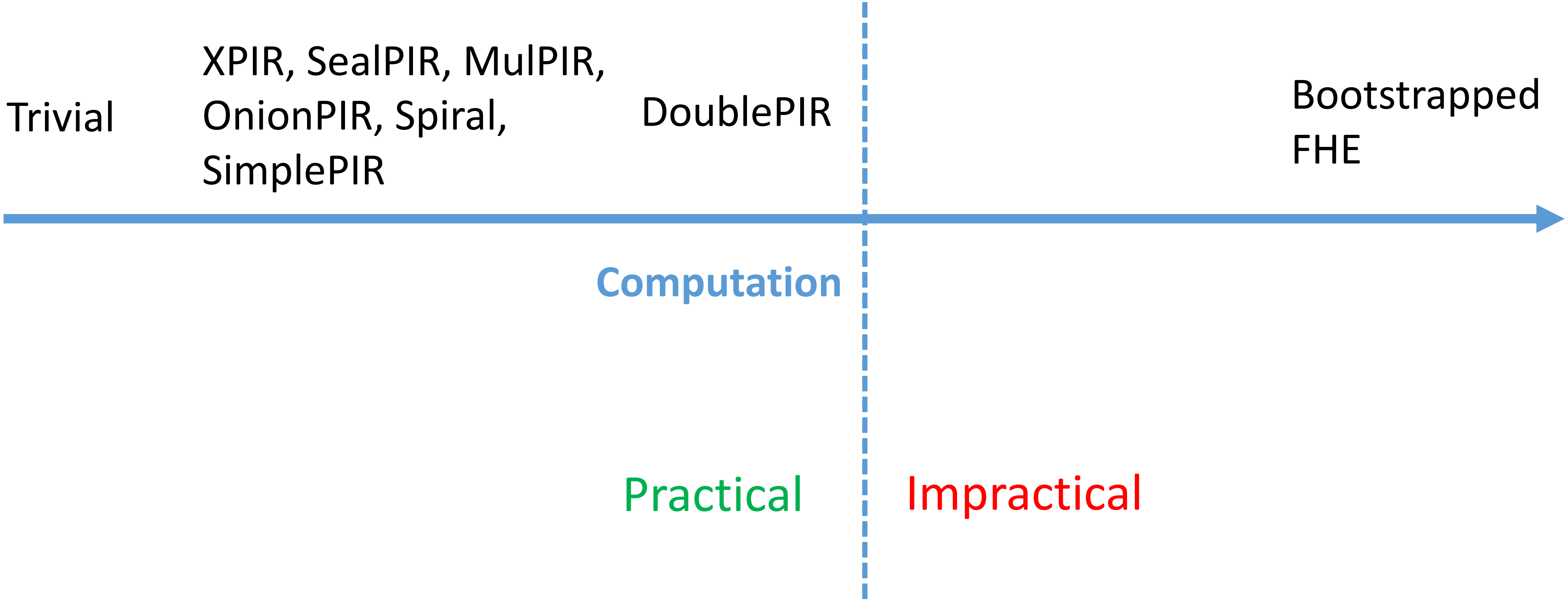
Communication: $\mathbf{O(\sqrt{n})}$

Computation: *More*

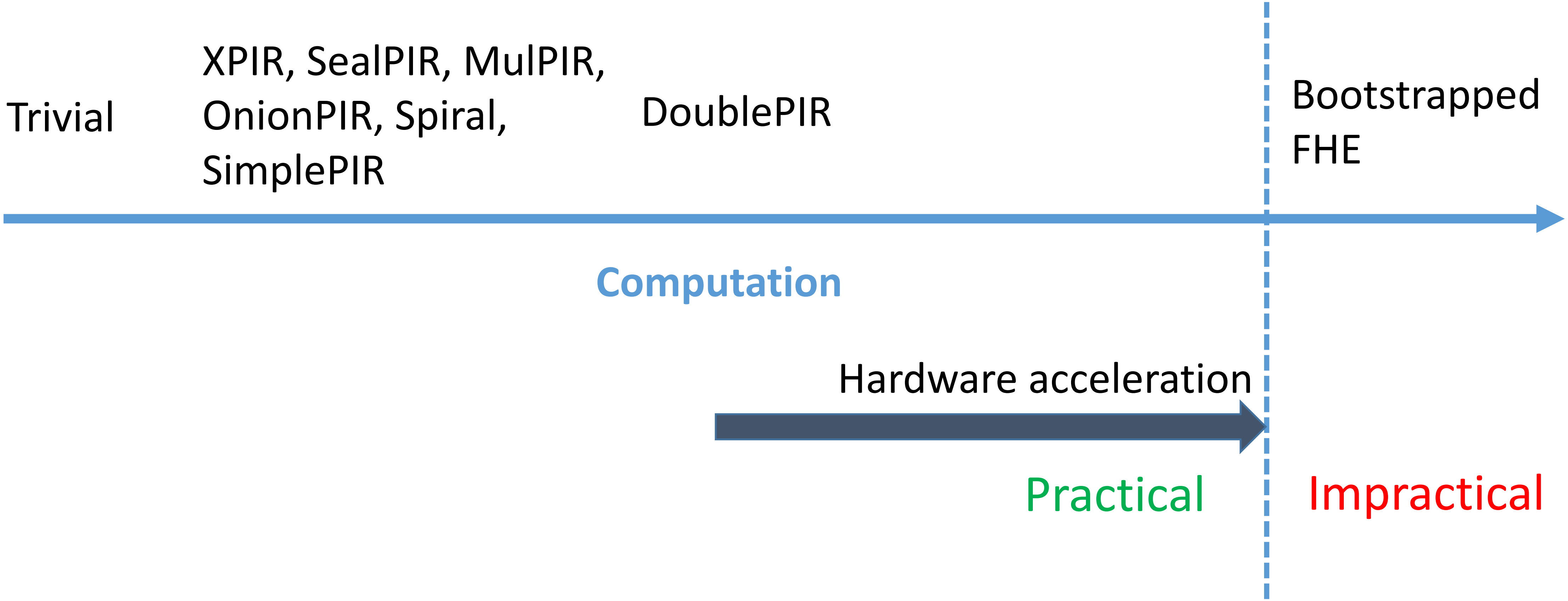
Communication: $\mathbf{O(\sqrt[3]{n})}$

Computation: *A lot!*

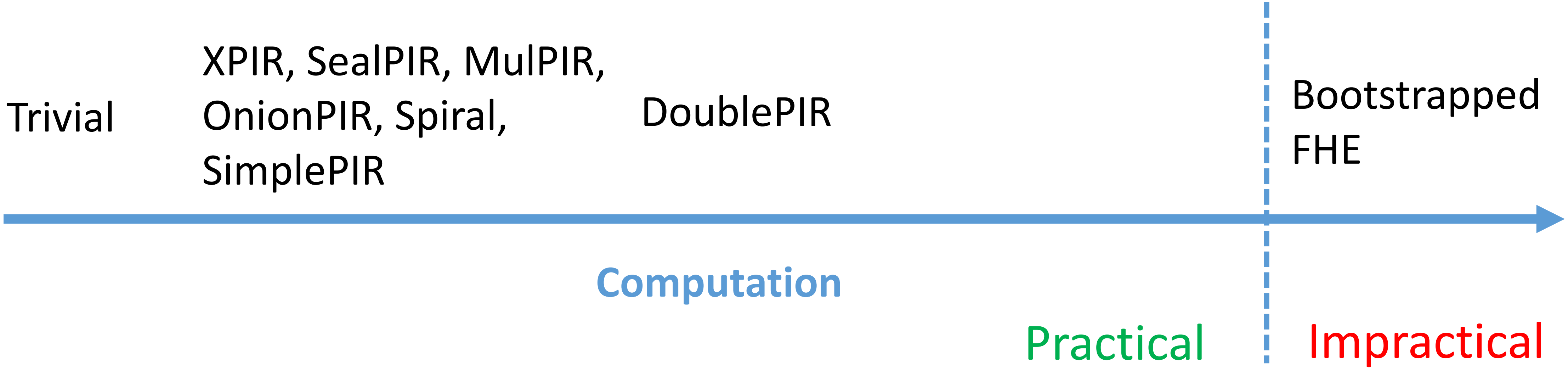
PIR Tradeoffs



PIR Tradeoffs



PIR Tradeoffs



PIR (no acceleration):

- 300 KB of communication
- 600 seconds of computation

CPU bottlenecked

PIR Tradeoffs

F1 can accelerate these schemes

CraterLake can accelerate this

Trivial

XPIR, SealPIR, MulPIR, OnionPIR, Spiral, SimplePIR

DoublePIR

Bootstrapped FHE

Computation

Practical | Impractical

PIR (no acceleration):

- 300 KB of communication
- 600 seconds of computation

PIR with acceleration (estimate):

- 300 KB of communication
- A second of computation

CPU bottlenecked → Acceleration → Network / memory bottlenecked

Outline

What can hardware do for security?

Accelerating cryptography

- Case study: Fully Homomorphic Encryption (FHE)

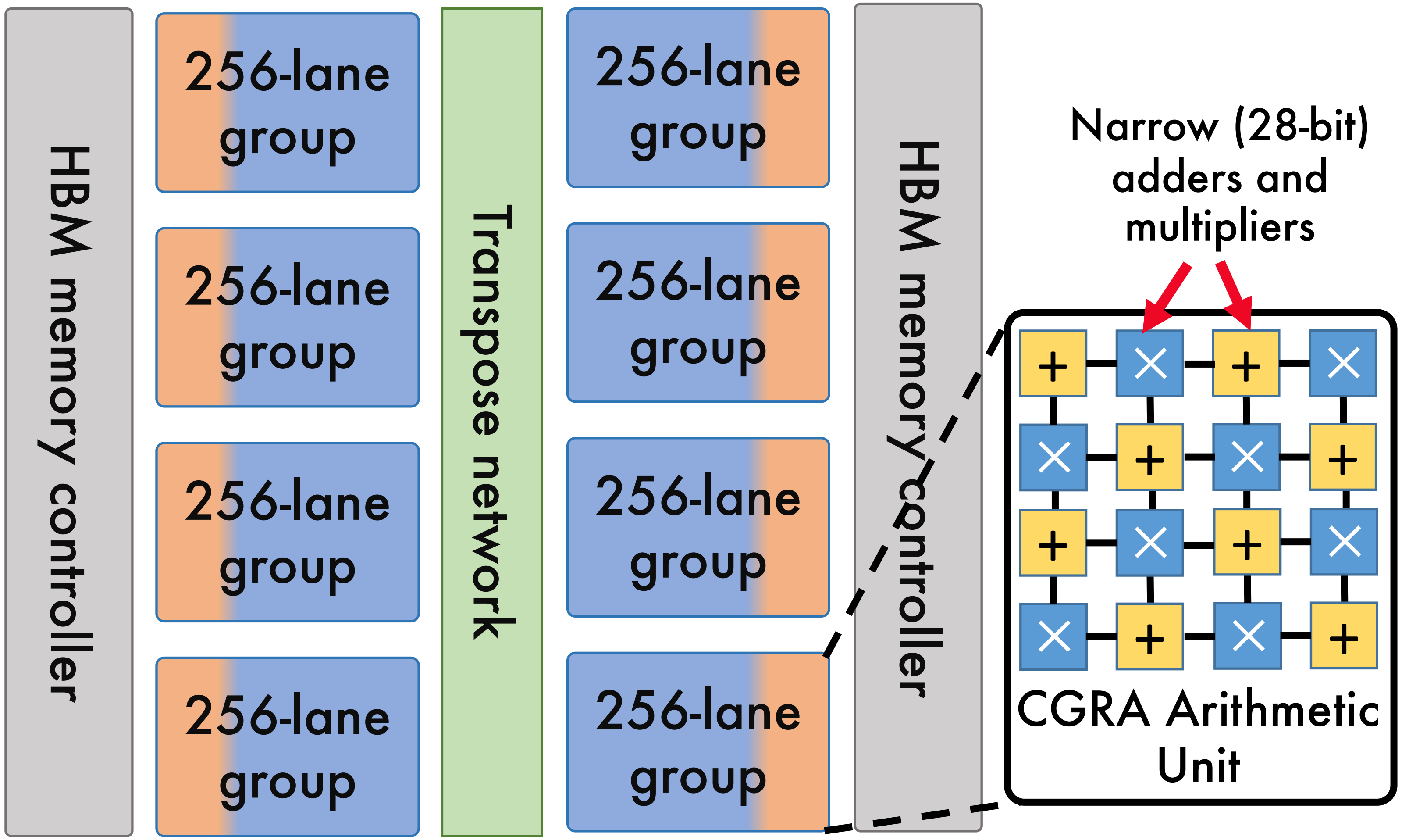
Should cryptography change with hardware accelerators?

What's next?

FHE + VC for Privacy + Integrity?

- Directly compose FHE and VC protocols: Two options
  - General VC system to prove that all of the server's FHE operations were computed correctly
  - The entire VC proof generation could be evaluated inside of FHE – only has covert security
- The overhead of FHE is multiplied by the overhead of VC, resulting in a wildly impractical protocol
- Hardware acceleration seems to only apply to the outermost protocol, not the other

A Universal Cryptographic Accelerator?



Thank You!