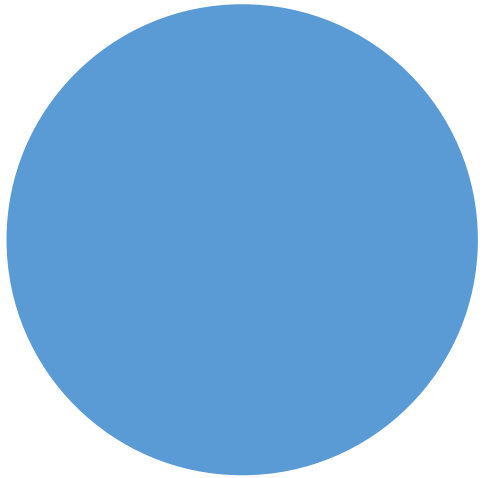
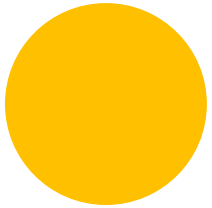



Cryptographic Computations Need Compilers

Madan Musuvathi
Microsoft Research

Microsoft's Commitment to Research

~1000 People **and growing!**

ca. 650 researchers

ca. 350 engineers, testers, designers, PMs

Labs around the World

Bangalore, Beijing, Cambridge (MA), Cambridge (UK), New York, Redmond

Sponsor of Conferences and Academic Research

Biggest PhD Internship Program

RISE

Research in Software Engineering

Programming Languages

- Programming Models
- HPC, Compilers, Systems
- Verified Programming

Software Engineering

- Reliability Tools
- Productivity
- SE4AI & AI4SE

Automated Reasoning

- Foundations
- Theorem Proving

Research in Software Engineering (RiSE)


Tom Ball


Christian Bird


Nikolaj Bjorner


Ella Bounimova


Sebastian Burckhardt


Patrice Godefroid


Peli de Halleux


Markus Kuppe


Shuvendu Lahiri


Daan Leijen


Saeed Maleki


Mark Marron


Kenneth McMillan


Michal Moskal


Leonardo de Moura


Madan Musuvathi


Todd Mytkowicz


Lev Nachmanson


Nachi Nagappan


Jonathan Protzenko


Tahina Ramananandro


Olli Saarikivi


Nikhil Swamy


Margus Veanes


Tom Zimmermann


Ben Zorn

RiSE Hires 2019

Daniel Selsam

Ph.D. Stanford University
Formal Methods & Machine Learning

Denae Ford Robinson

Ph.D. NC State University
HCI & Software Engineering

Olli Saarikivi

Ph.D. Aalto University
Compiler Systems

Teddy Seyed

Ph.D. University of Calgary
HCI & Embedded Systems

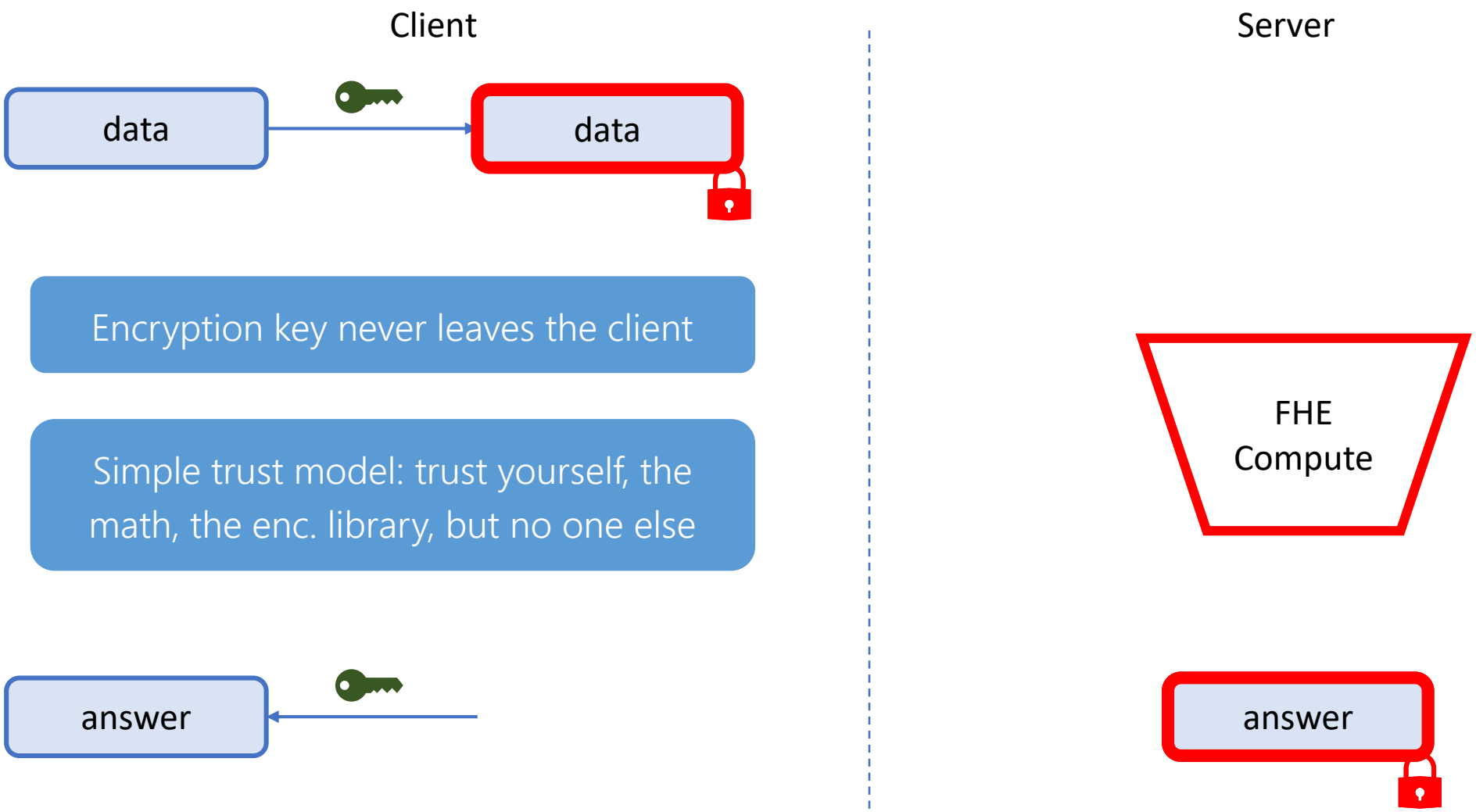
My background

- Combine formal methods and systems
- Model checking real systems [OSDI ’02, NSDI ‘04, OSDI ‘04]
- Finding concurrency bugs in the large [PLDI ‘07, OSDI ‘08, ASPLOS ’10, OSDI ’10, SOSP ‘19]
- Memory consistency models [CAV ’08, POPL ‘10, PLDI ‘10, PLDI ‘11, ISCA ’12, OOPSLA ‘17, PPoPP ‘17, PLDI ‘19]
- Parallelizing seemingly sequential computations [ASPLOS ‘14, PPoPP ‘14, SOSP ‘15, ICASSP ‘16, IPDPS ‘18]

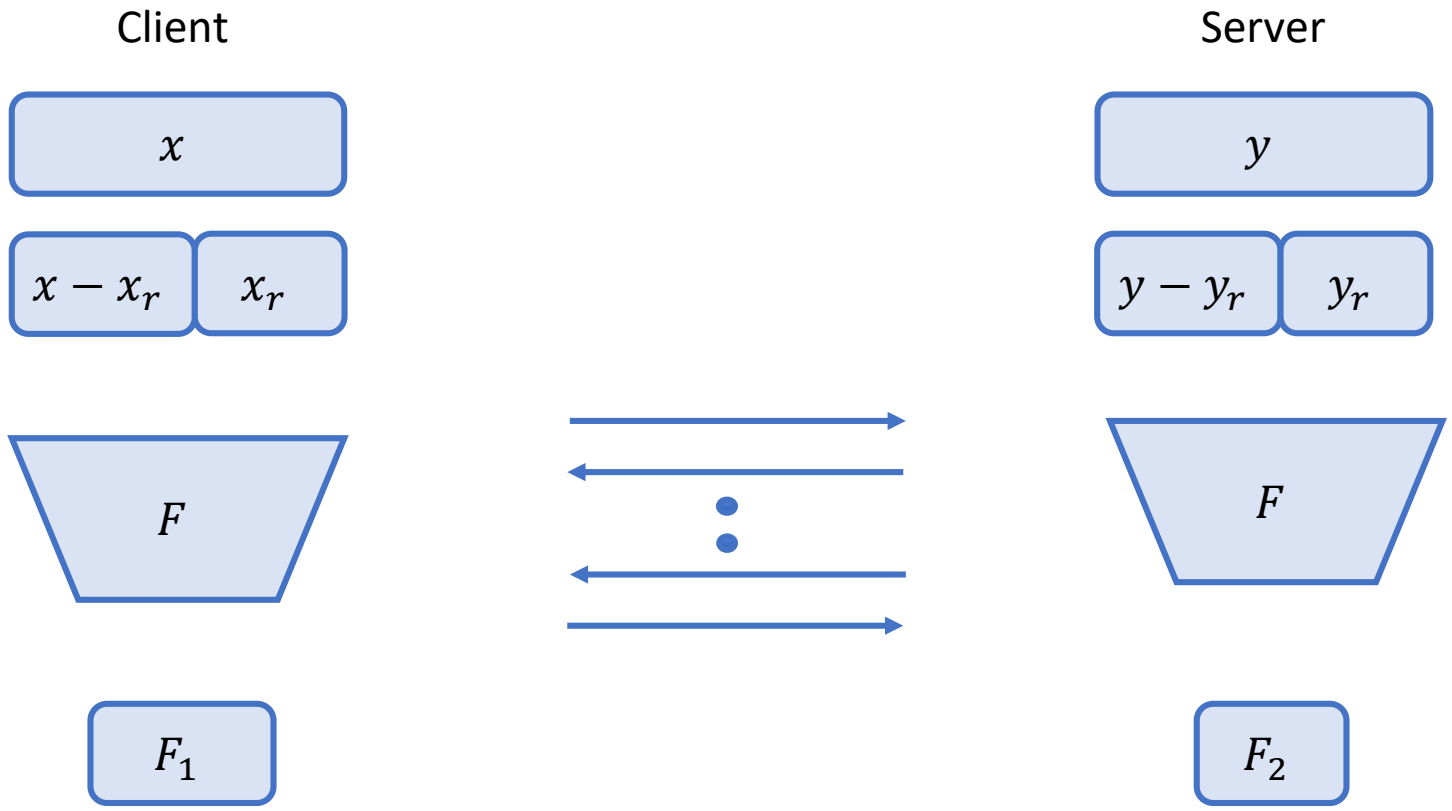
Cryptographic Computations Need Compilers

Jointly with
Roshan Dathathri (University of Texas, Austin)
Betty Pirelli (EPFL)
Olli Saarikivi, Saeed Maleki, Todd Mytkowicz (MSR RiSE)
Hao Chen, Wei Dai, Kim Laine, Kristin Lauter (MSR Crypto)

Fully Homomorphic Encryption (FHE)



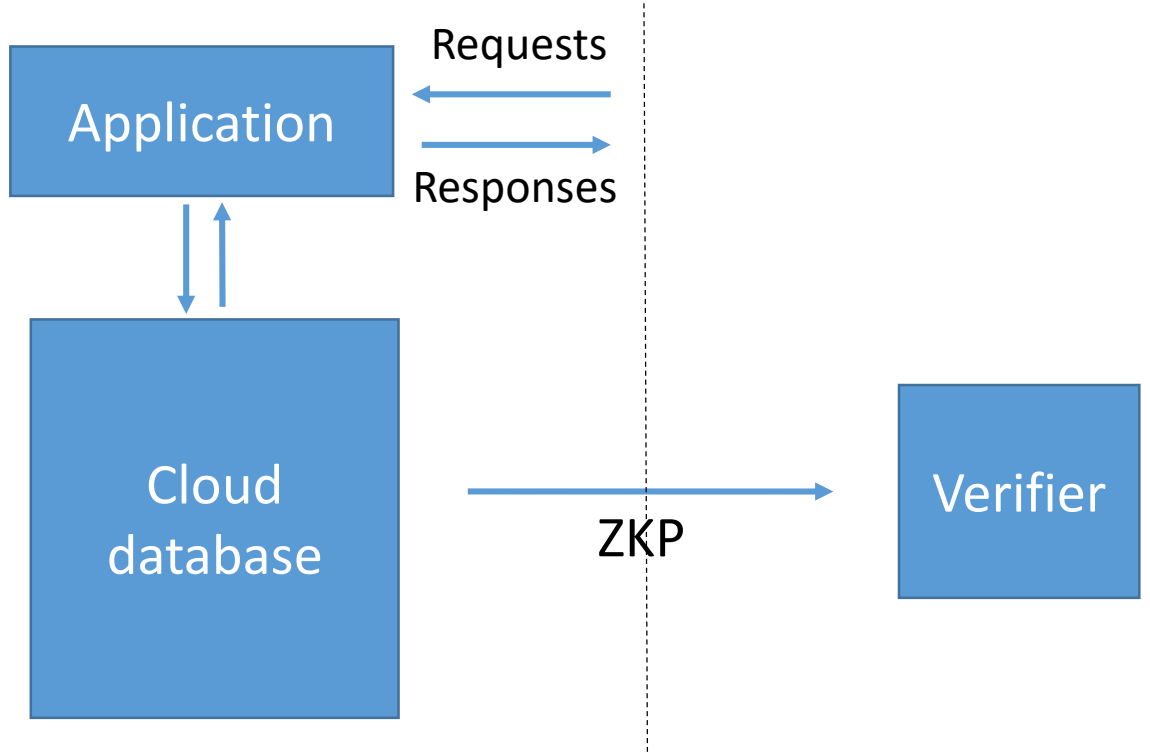
Secure Multi-Party Computation (Secure MPC)



Nothing is leaked except the output

$F(x, y)$

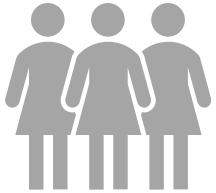
Zero Knowledge Proofs (ZKP)



Cryptographic proof that desired code was correctly executed without leaking requests, responses, or state

Cryptographic Computations need Compilers

Fully Homomorphic Encryption



Secure Multi-Party Computation

Zero Knowledge Proofs

Convert a (bounded) program into a circuit

Apply domain-specific optimizations

Guarantee correctness, security, and efficiency

Fully-Homomorphic Encryption (FHE)

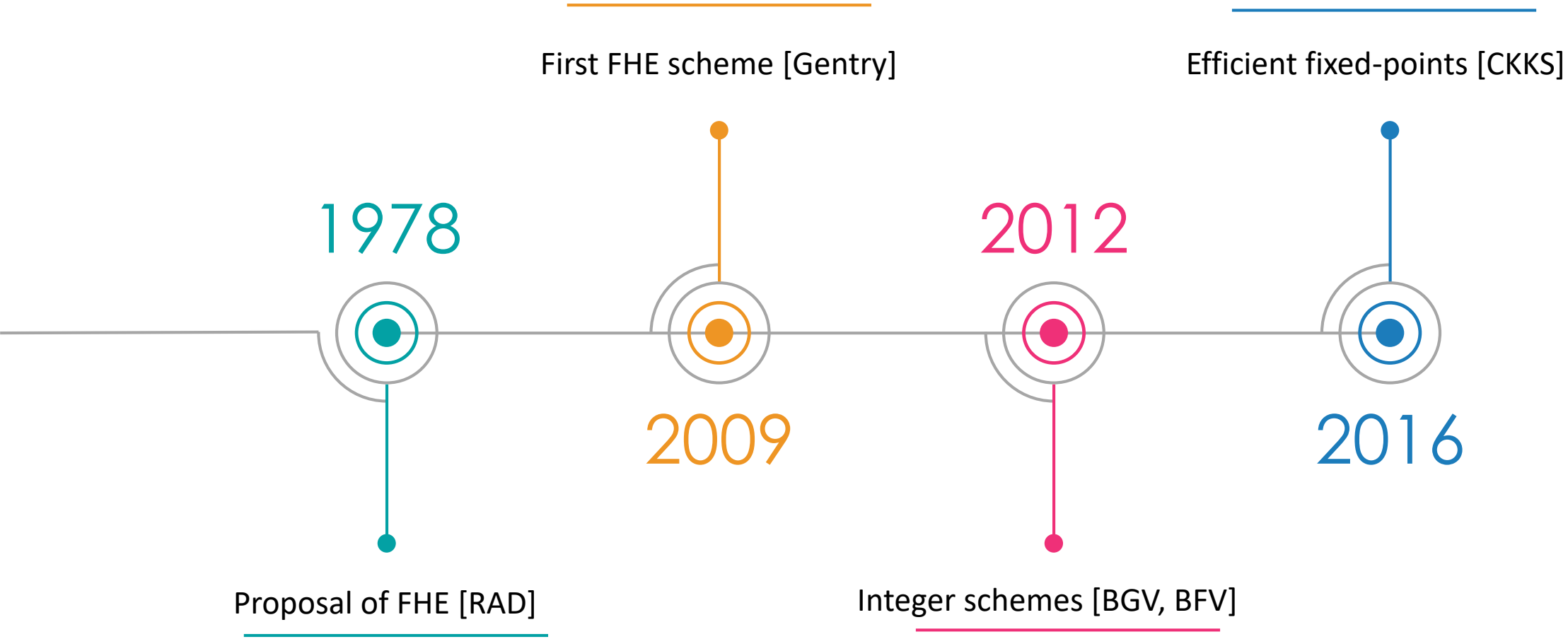
Allows computation on encrypted data

$$[\![m]\!] \triangleq Enc(m)$$

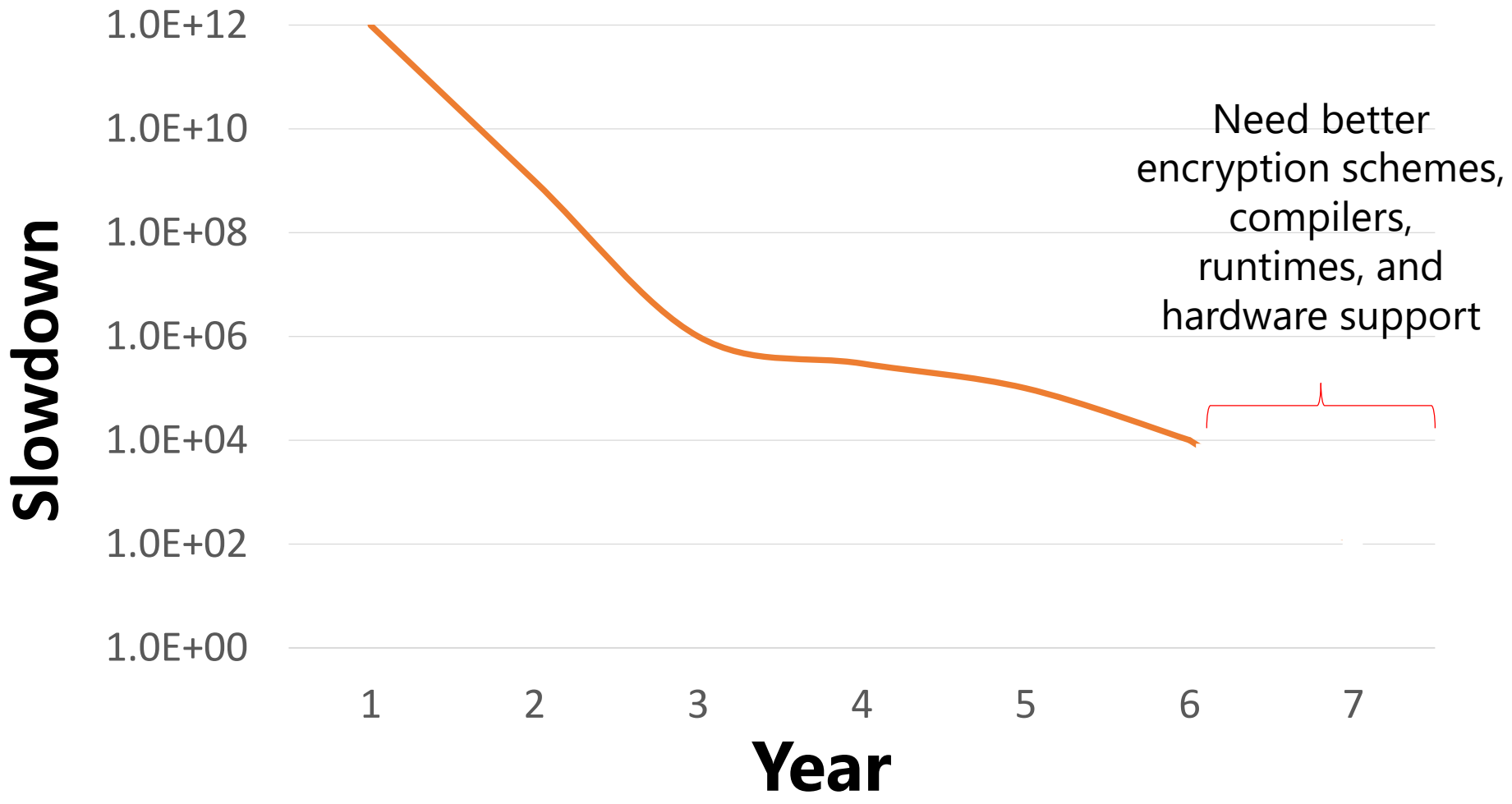
$$[\![a]\!] \oplus [\![b]\!] = [\![a + b]\!]$$

$$[\![a]\!] \otimes [\![b]\!] = [\![a \times b]\!]$$

FHE timeline



Performance overhead of FHE over unencrypted



FHE programming challenges

Computation is slow: ~50 ms per multiplication

Very large SIMD vector widths: ~16K
- can operate at 8086 speeds if you can utilize the parallelism

No branching
- a bug and a feature

Tradeoff between correctness, security, message bloat, and performance
- requires setting parameters correctly

Different encryption schemes provide different functionalities
- arithmetic, Boolean logic, table lookups, …

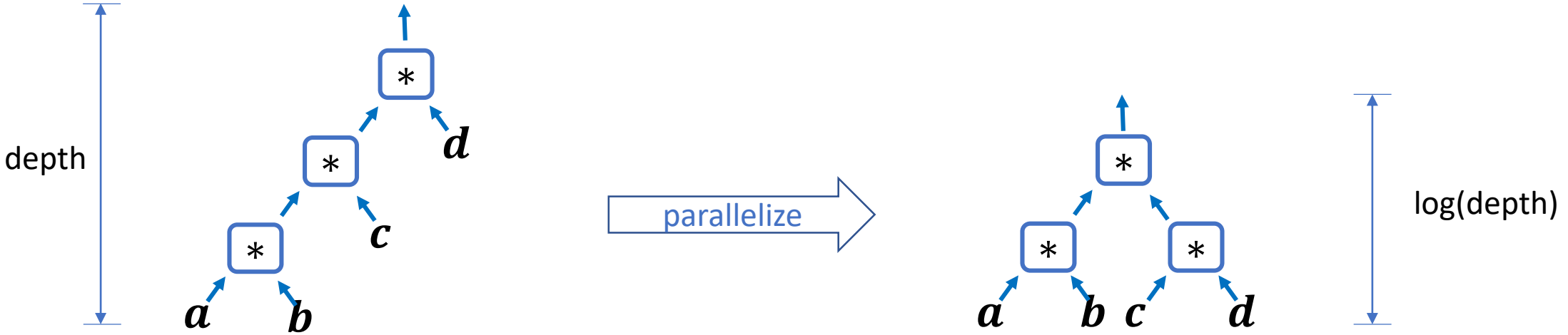
Cryptographic noise restricts multiplicative depth of circuits

Noise growth challenge

Noise growth proportional to *multiplicative depth*

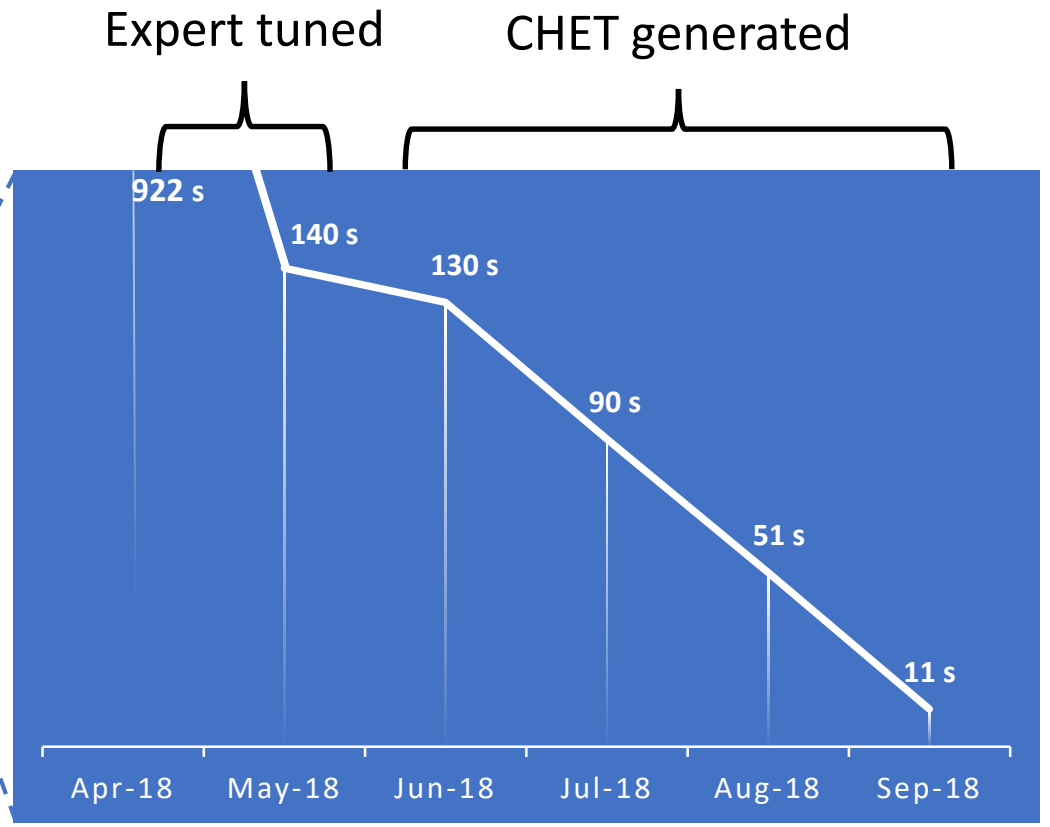
$$Noise([\![a \times b]\!]) = \max(Noise([\![b]\!]), Noise[\![a]\!]) + k$$

Need expensive *bootstrapping* after ~20 multiplications

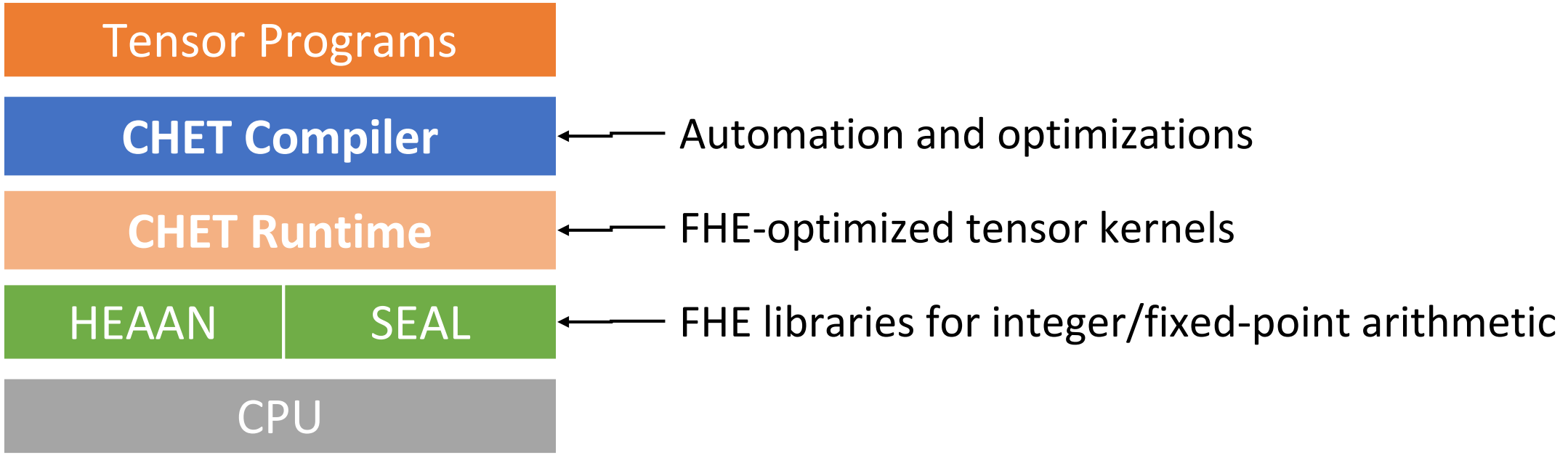


CHET [PLDI ’19] FHE DNN Inference Latency

Network	CHET
LeNet5-small	3 s
LeNet5-medium	11 s
LeNet5-large	35 s
Industrial	56 s
SqueezeNet-CIFAR	165 s

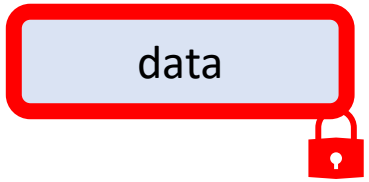


Compiling Tensor Programs for FHE libraries



Encryption Parameters

Ciphertext is a high-degree polynomial



$$= a_N \cdot x^N + a_{N-1} \cdot x^{N-1} + \cdots a_1 \cdot x + a_0$$

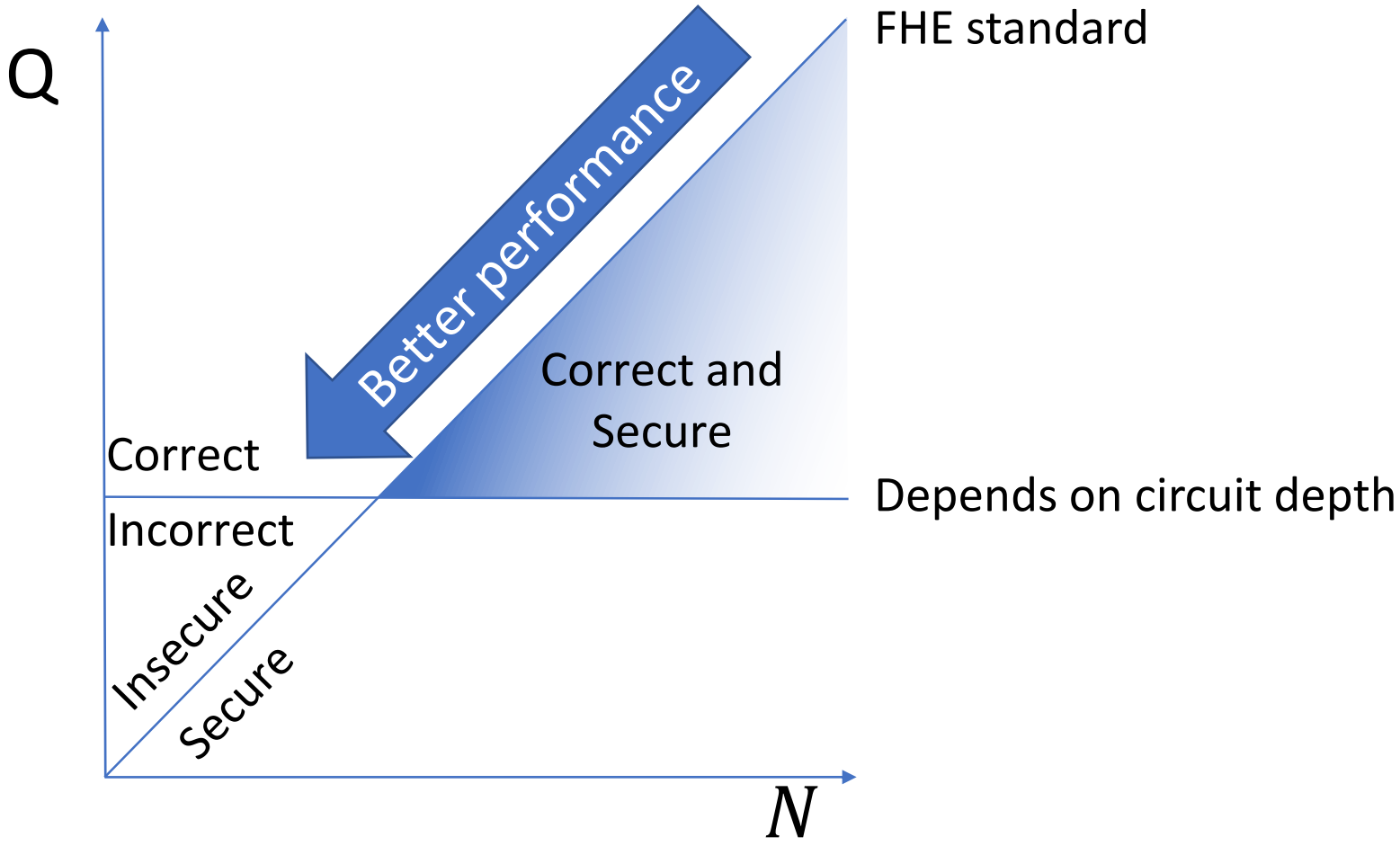
N: degree of the polynomial

Q: modulus of the coefficients

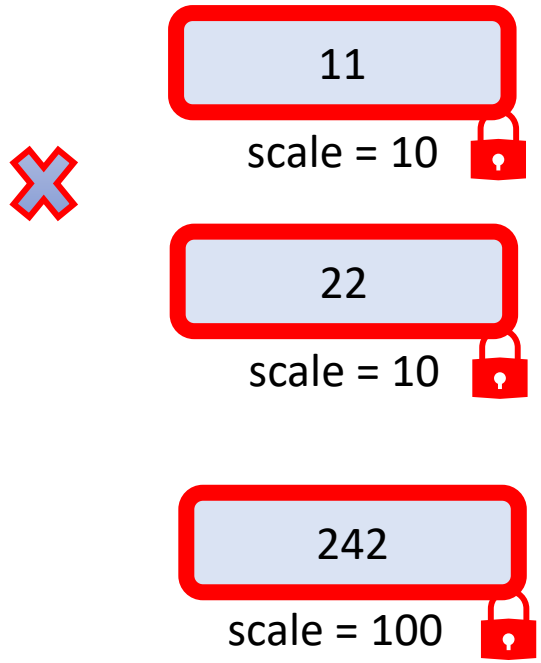
Encryption Parameters

N: degree of the polynomial

Q: modulus of the coefficients



Performing Fixed Points with Scaling Factors



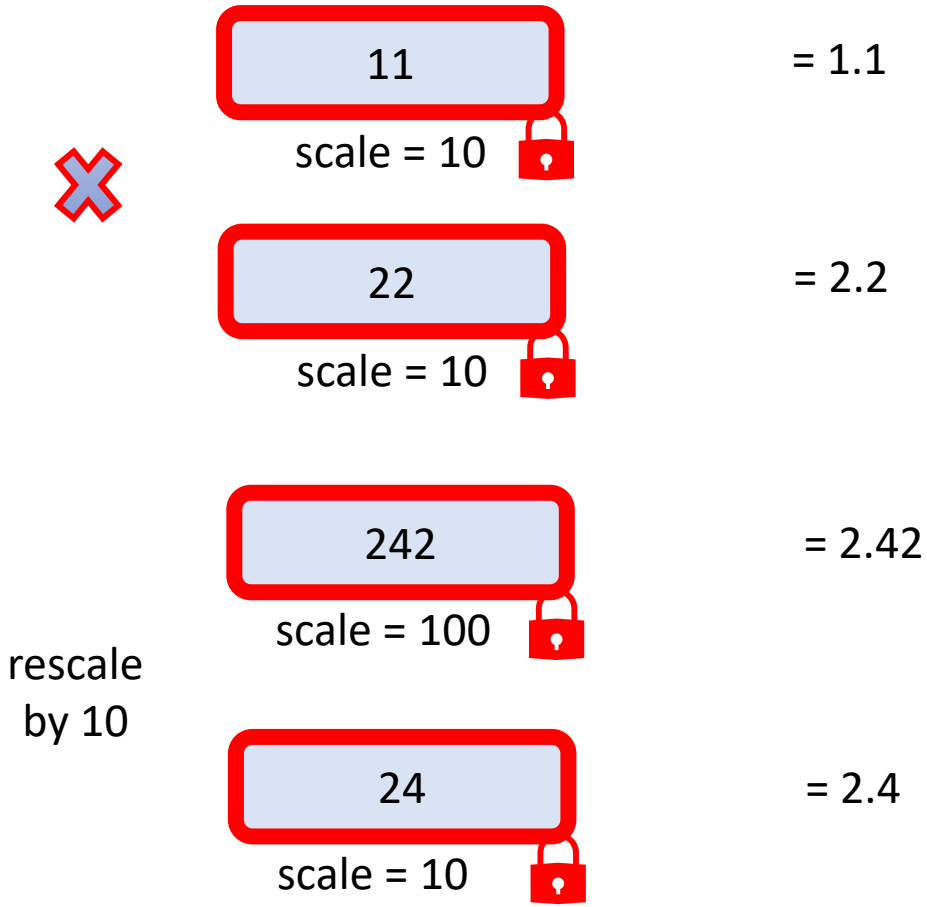
= 1.1

= 2.2

= 2.42

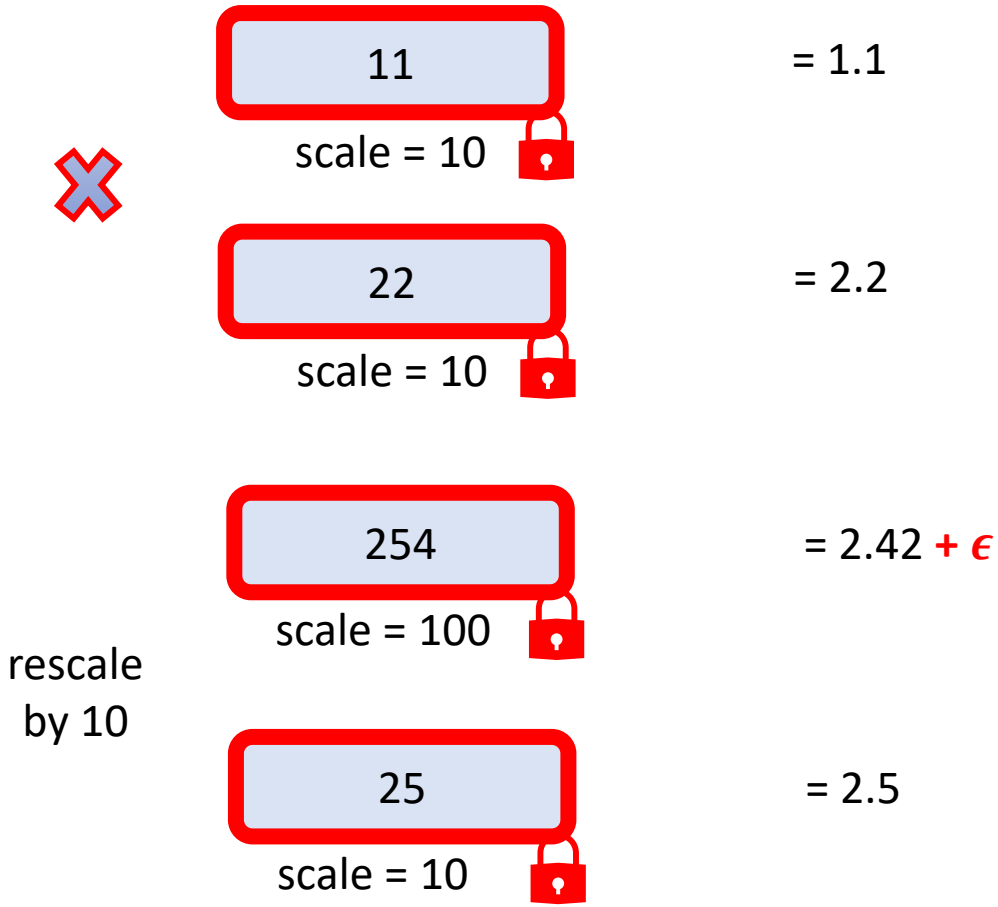
Modulus growth limits depth of circuits

Rescaling operation in CKKS ‘16

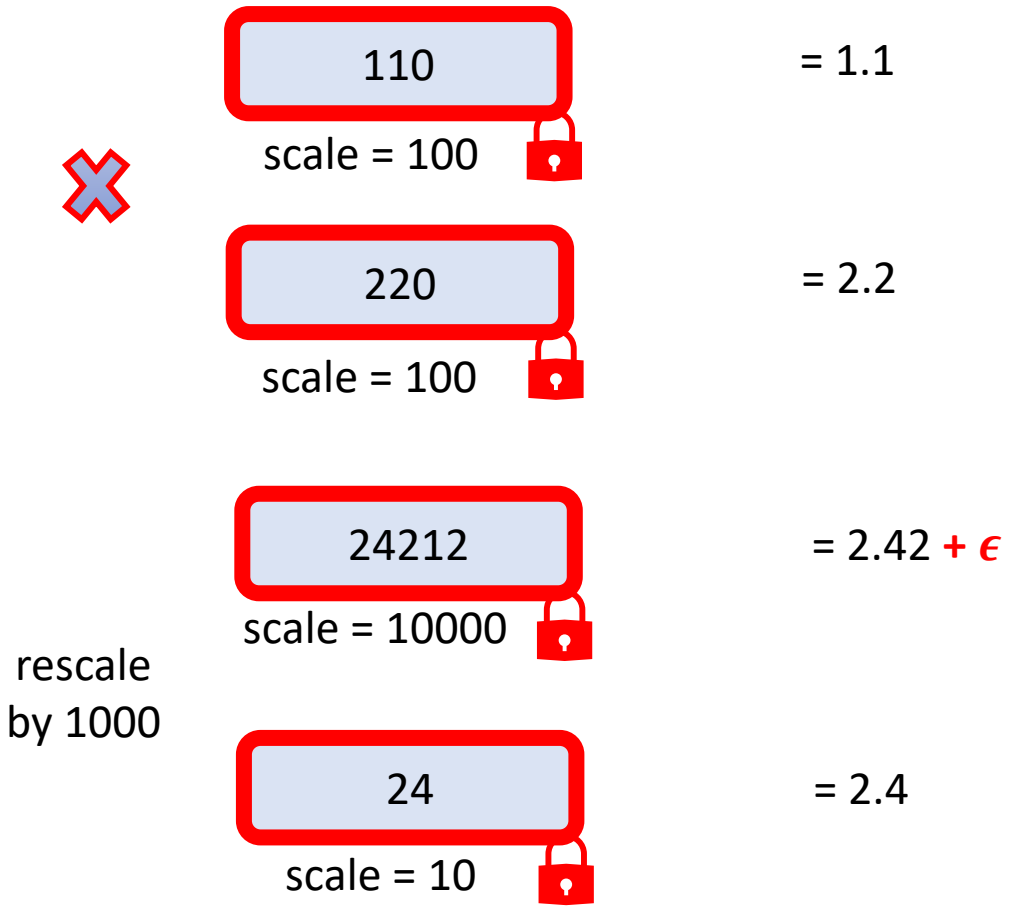


Compiler needs to insert
rescale operations effectively

But, CKKS is approximate



Solution: inflate scale

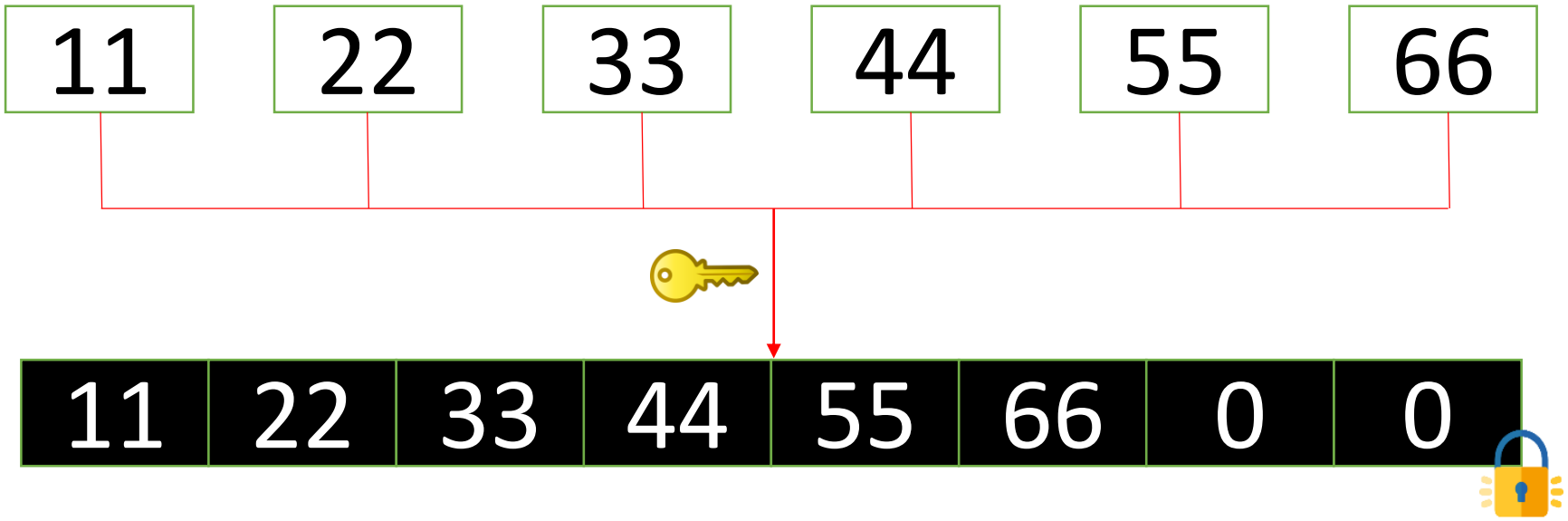


Compiler needs to manage precision and error

FHE Packing

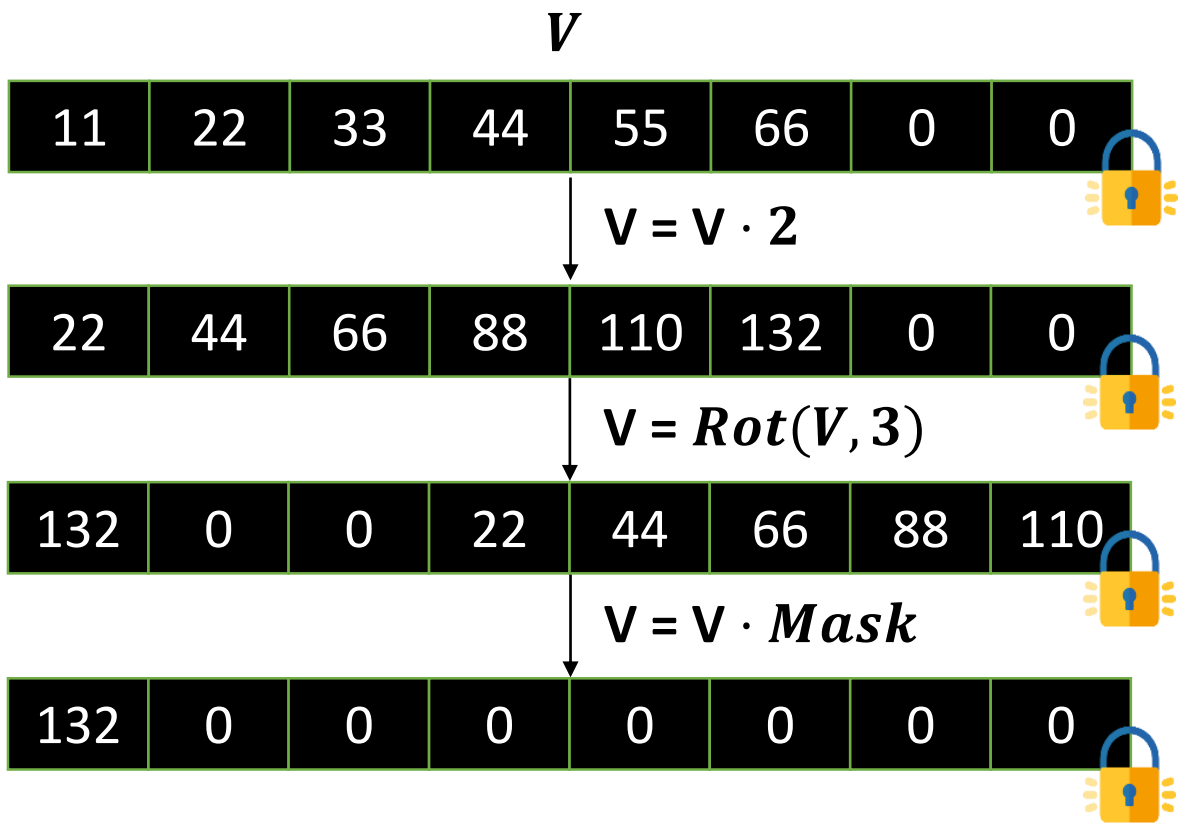
Pack many plaintext scalars into single ciphertext vector

- Fixed width of N/2 (N is 2^{10} or larger)

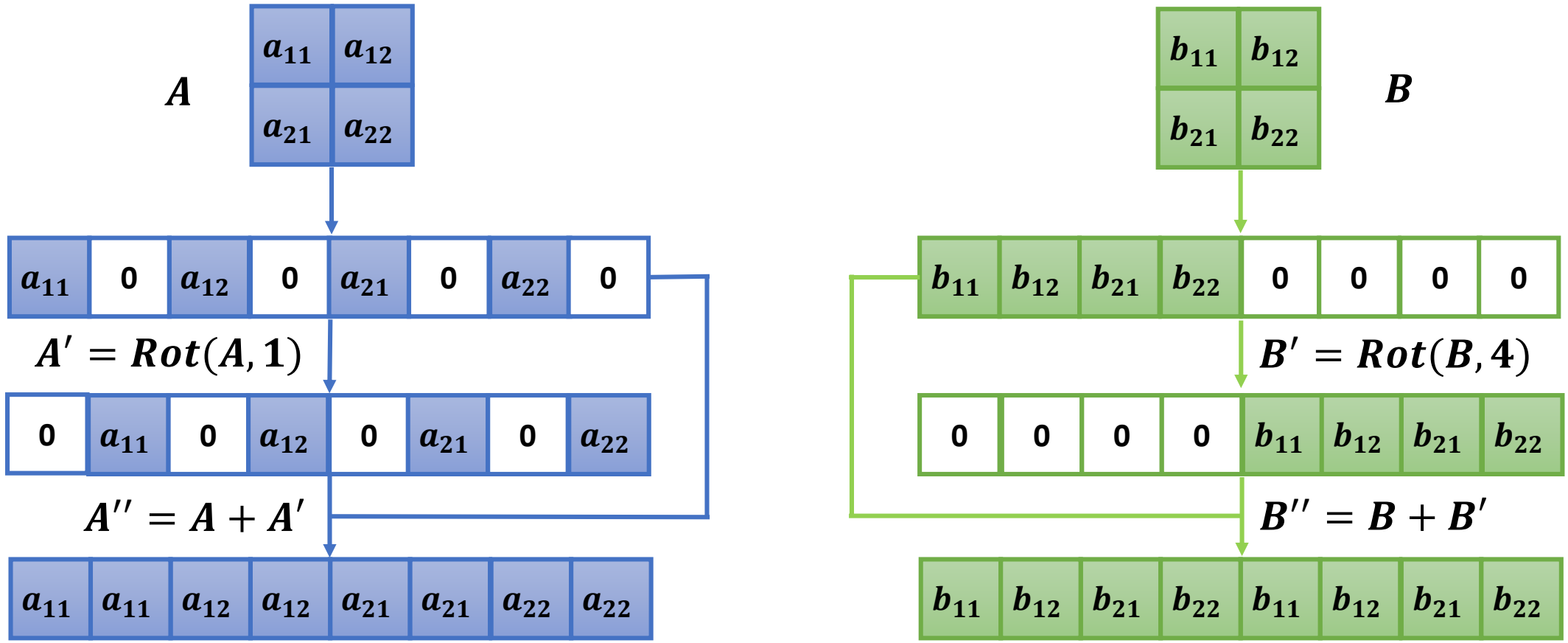


FHE Vectorization

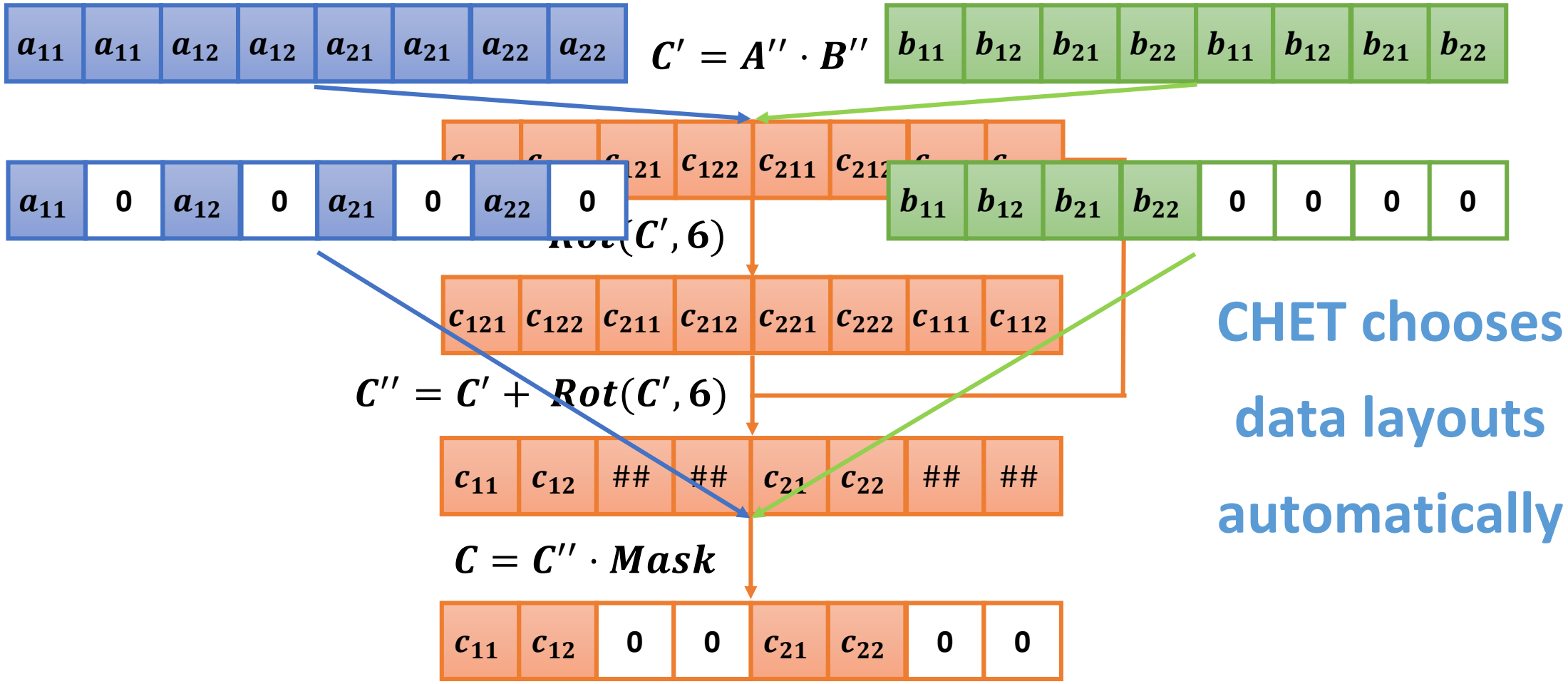
- **Limited set of SIMD instructions**:
  - Element-wise addition
  - Element-wise subtraction
  - Element-wise multiplication
  - Rescale (all elements)
  - Rotation
- Random access of a vector element not supported



Example of Matrix Multiplication: C = A x B

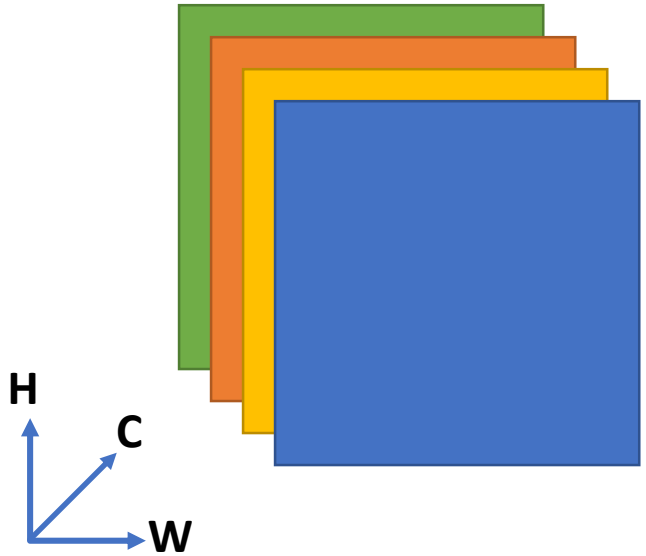


Example of Matrix Multiplication: C = A x B

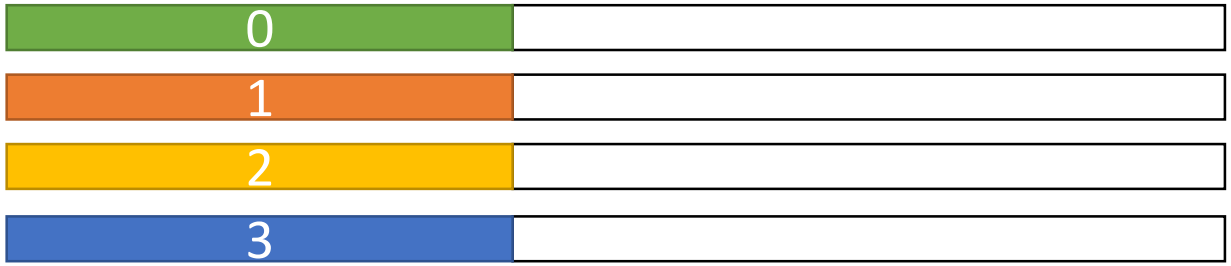


CHET chooses data layouts automatically

Mapping Tensors to Vector of Vectors



There are many ways to layout tensors into vectors, with different tradeoffs



HW (Height-Width) layout:

- Easier convolutions due to channels being aligned
- Wasted space

CHW (Channel-Height-Width) layout:

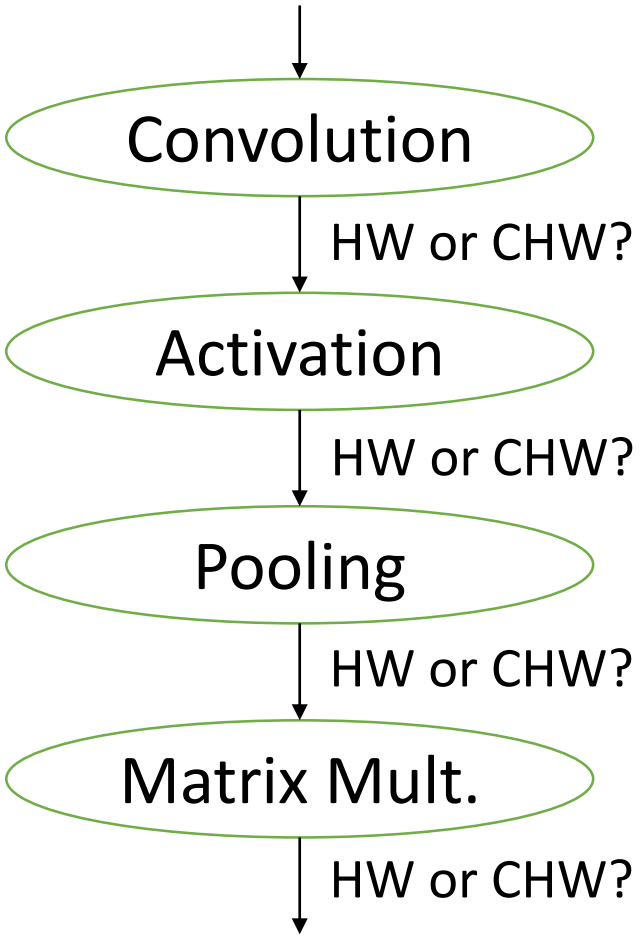
- More efficient space usage
- Convolutions require more rotations

Data Layout Selection

- Search space: explore possible data layouts
- Cost estimation: estimate cost of each search point
- Pick the best-performing one

Data Layout Selection: Search Space

- Search space is exponential
  - 2 choices per tensor operation: **HW** or **CHW**
- Prune search space: use domain knowledge -> limits to only 4 choices for the circuit
  - Convolution faster in **HW** while rest faster in **CHW**
  - Matrix multiplication faster if output is in **CHW**



Data Layout Selection: Cost Estimation

- Cost model for FHE primitives:
  - Asymptotic complexity (specific to FHE scheme)
  - Microbenchmarking to determine constants
- Cost of a circuit: sum the costs for all operations

Experimental Setup

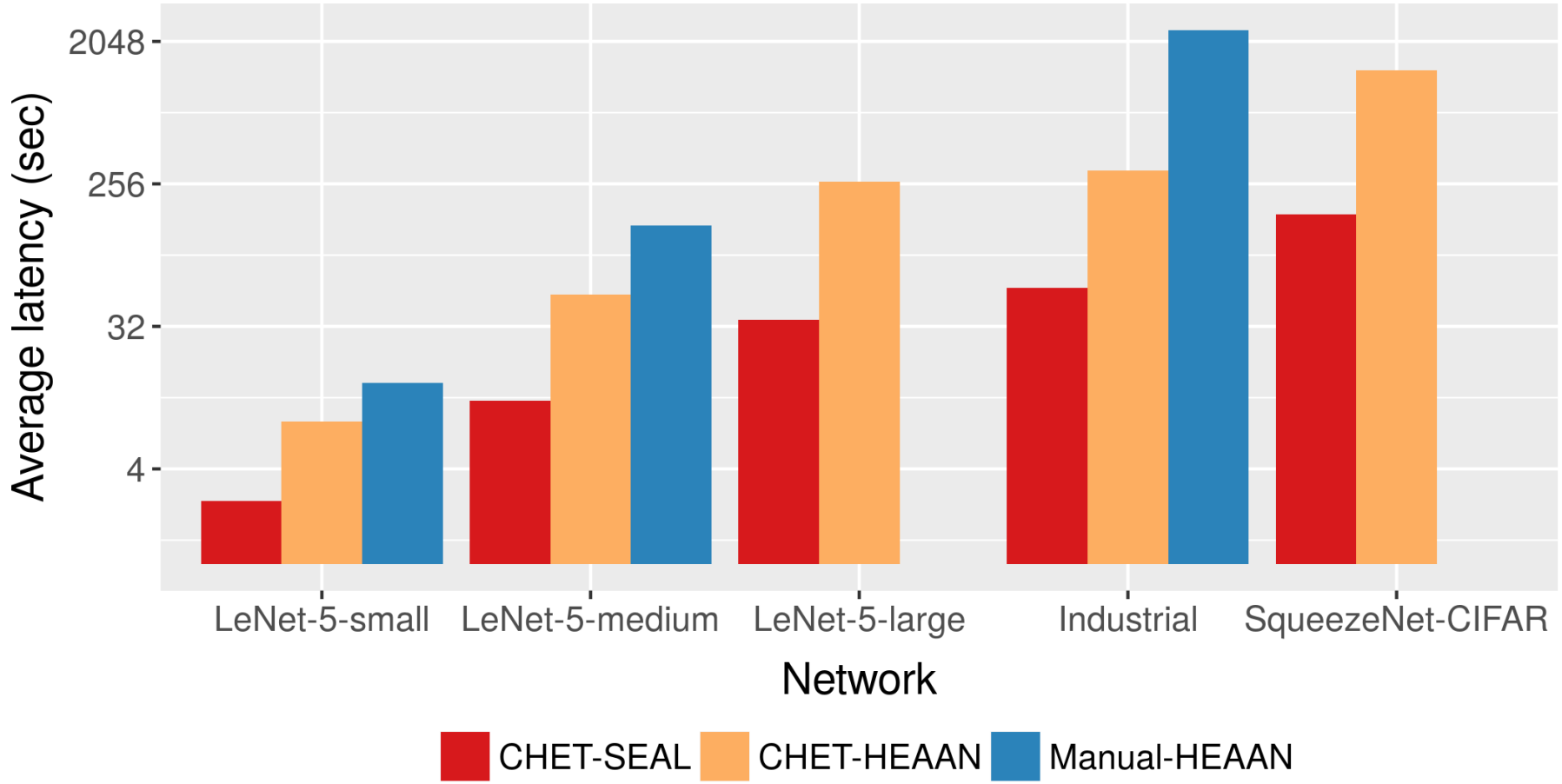
- **Systems:**
  - Hand-written HEAAN
  - CHET with HEAAN
  - CHET with SEAL
- **Machine:**
  - Dual-socket Intel Xeon E5-2667v3
  - 16 cores
  - 224 GB of memory

- **FHE-compatible Deep Neural Networks (DNN):**

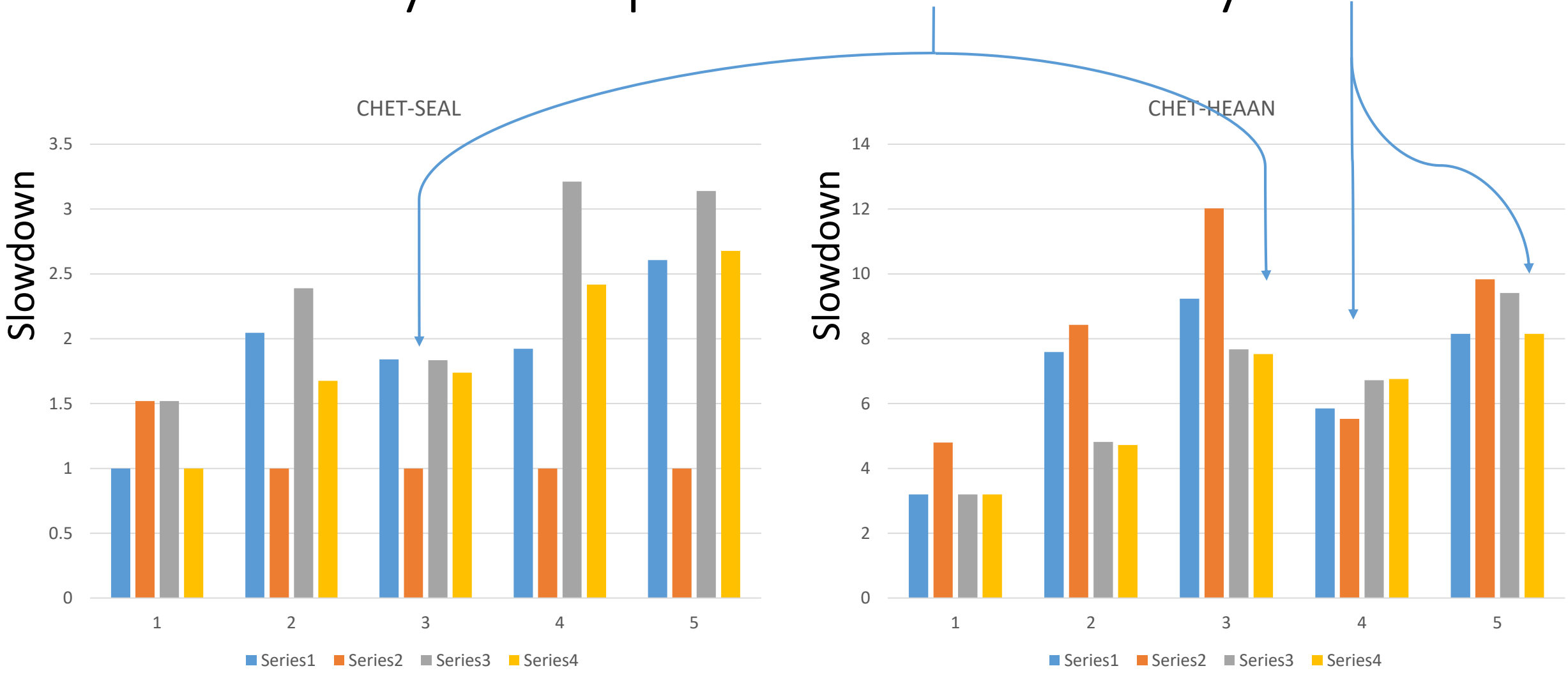
DNN	Dataset	# Layers	# FP ops (M)	Accuracy
LeNet-5-small	MNIST	8	0.2	98.5%
LeNet-5-medium	MNIST	8	5.8	99.0%
LeNet-5-large	MNIST	8	8.7	99.3%
Industrial	-	13	-	-
SqueezeNet-CIFAR	CIFAR-10	19	37.8	81.5%

- **Evaluation:**
  - Latency of image inference (batch size = 1)

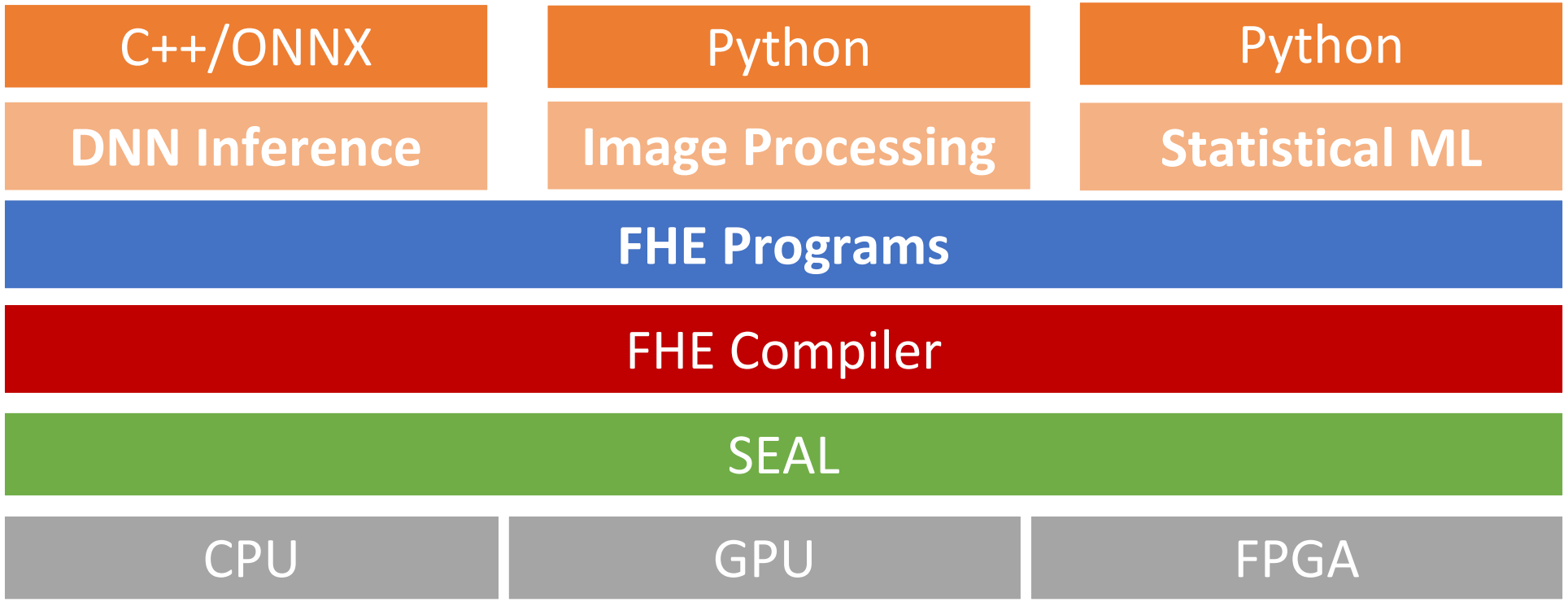
CHET outperforms hand-written implementations



Best data layout depends on FHE library and DNN

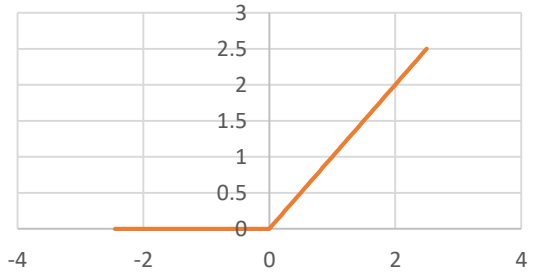


Generalizing CHET (ongoing work)

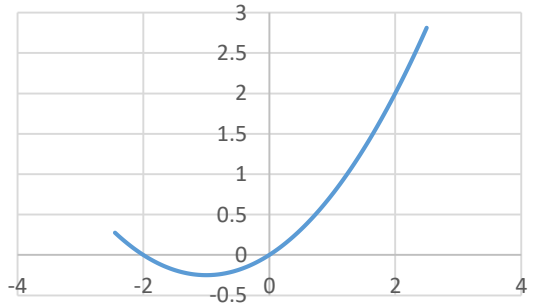


Making Computations FHE compatible

- Cannot evaluate non-polynomial operations
  - ReLU
  - Max pooling
- Options:
  1. Replace with polynomial approximation
  2. Combine Boolean and arithmetic schemes



$ReLU(x)$



$x^2 + a \cdot x$

Conclusions

- Cryptographic computation is a “PL + HPC + Systems” problem
- Currently at 8086 speeds but need two/three orders improvements
  - Better encryption schemes, compilers, runtime systems, HW support
- Interesting applications possible at current speeds
- Many PL challenges still open