ENCRYPTED COMPUTING SDK
POLYNOMIAL INSTRUCTION SET ARCHITECTURE TOOLS

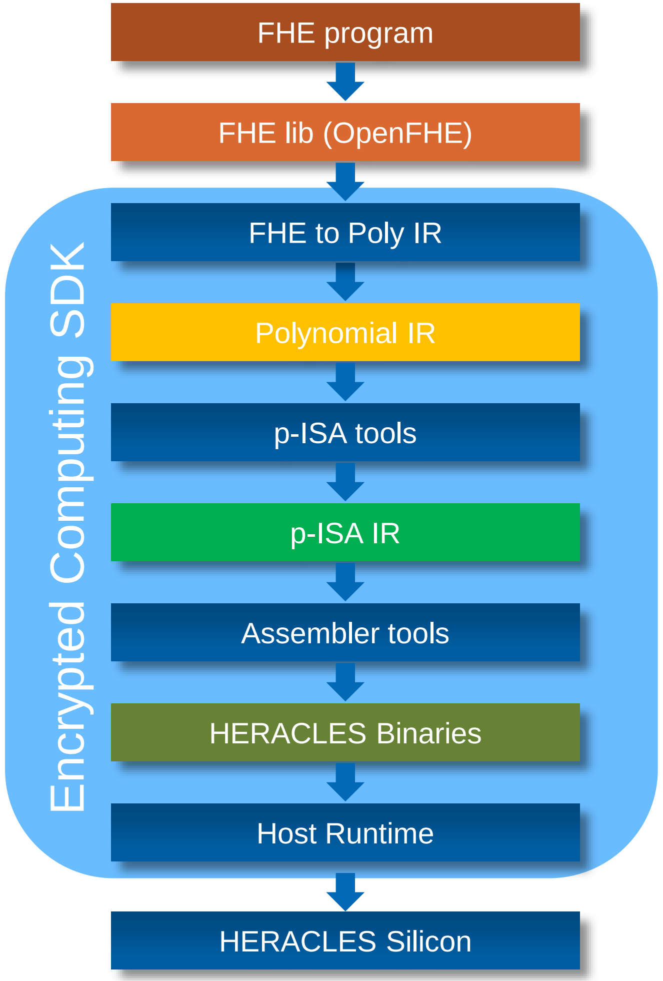
Flavio Bergamaschi, Privacy Technologies Research, Intel Labs

7th March 2025

Notices and Disclaimers

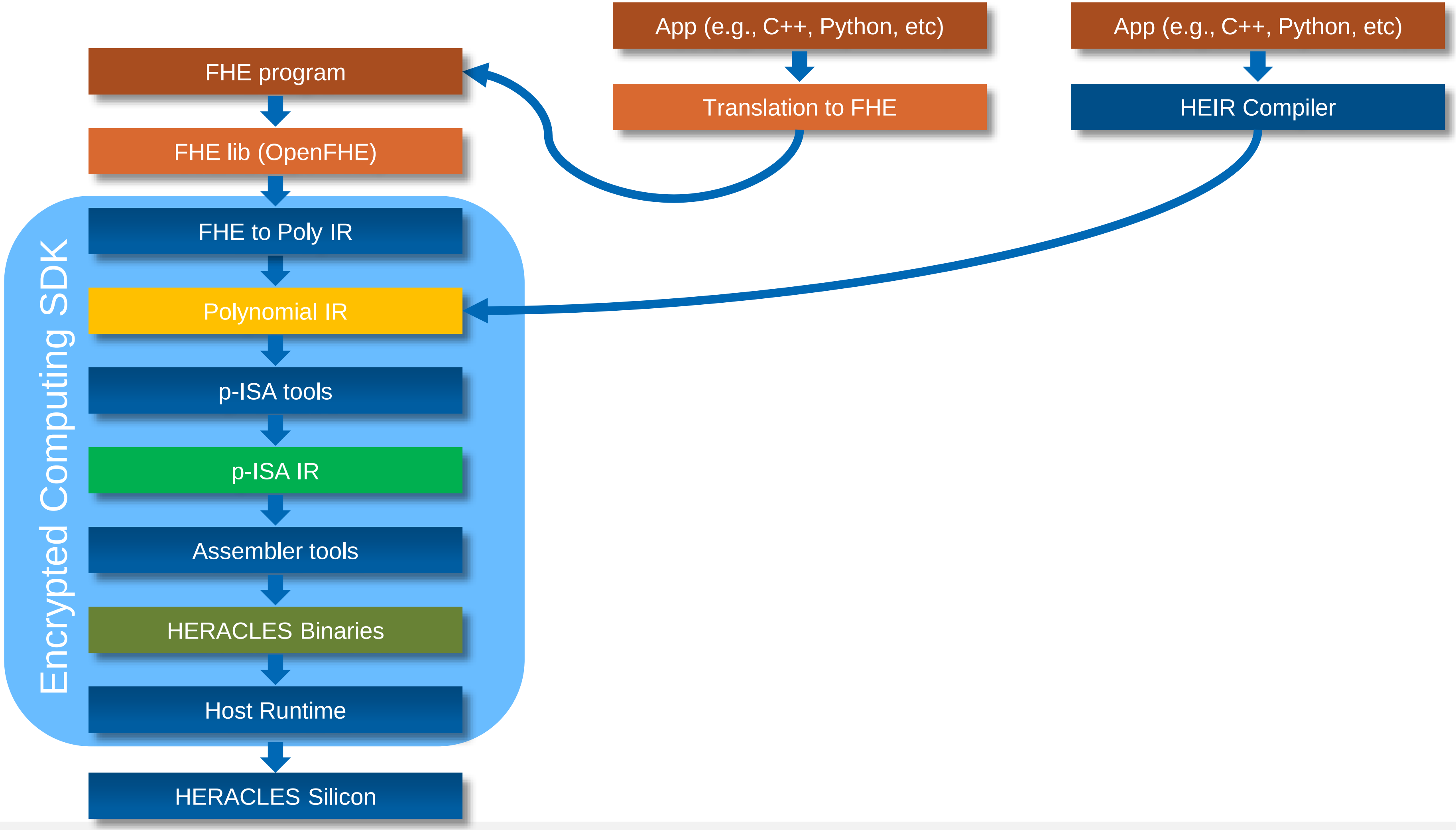
For notices, disclaimers, and details about performance claims, visit www.intel.com/PerformanceIndex or scan the QR code:

Encrypted Computing SDK Modular Approach



- Multistage transformation (compiler) pipeline
- Inspired by the LLVM
- Based on language independent intermediate representations (IR)
- Each stage promotes a separation of concerns
- Each stage applies dedicated transformations and optimizations

HERACLES SDK Integration with future 3rd Party Tools



FHE program

FHE lib (OpenFHE)

Encrypted Computing SDK

FHE to Poly IR

Polynomial IR

p-ISA tools

p-ISA IR

Assembler tools

HERACLES Binaries

Host Runtime

HERACLES Silicon

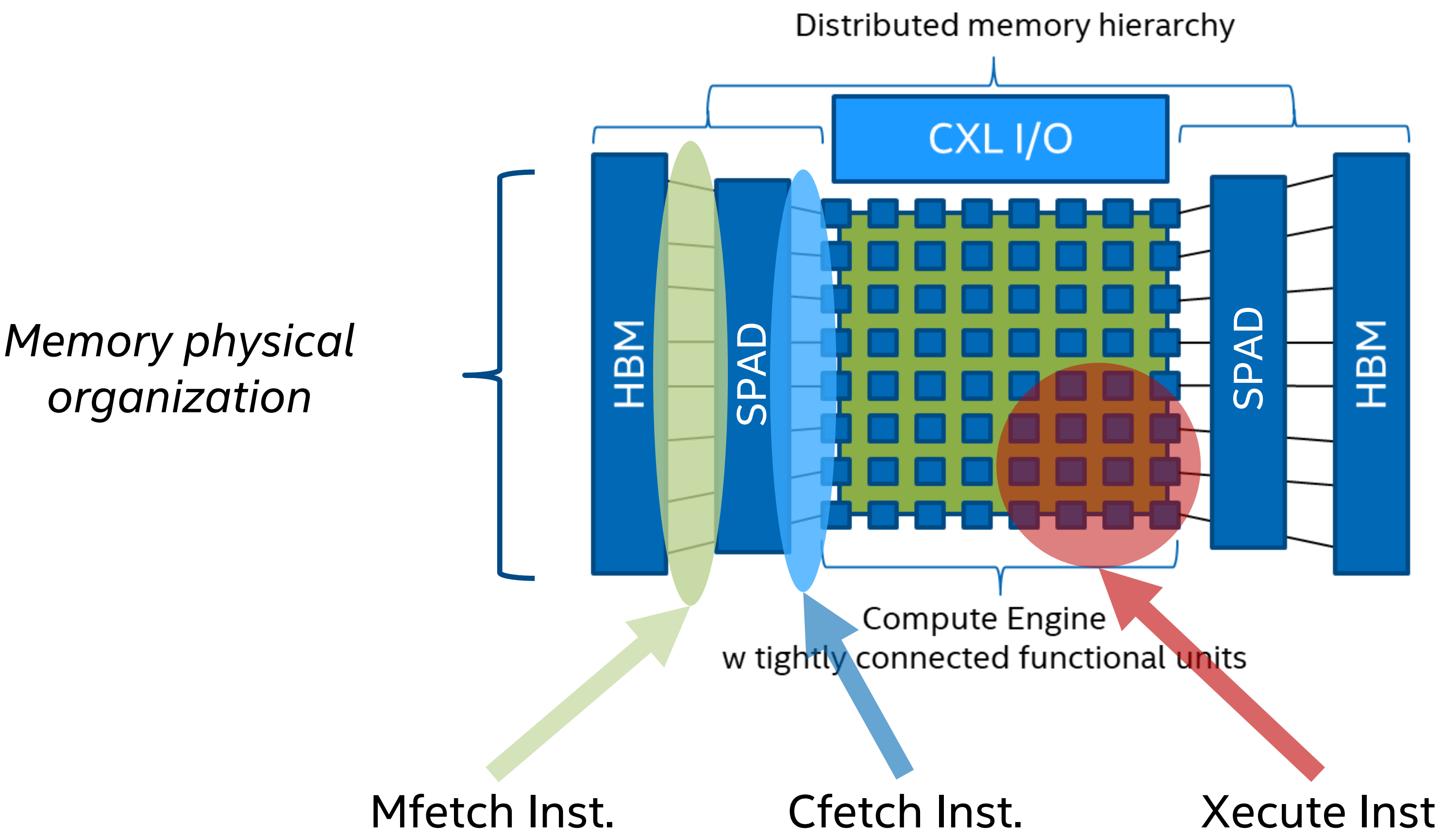
App (e.g., C++, Python, etc)

Translation to FHE

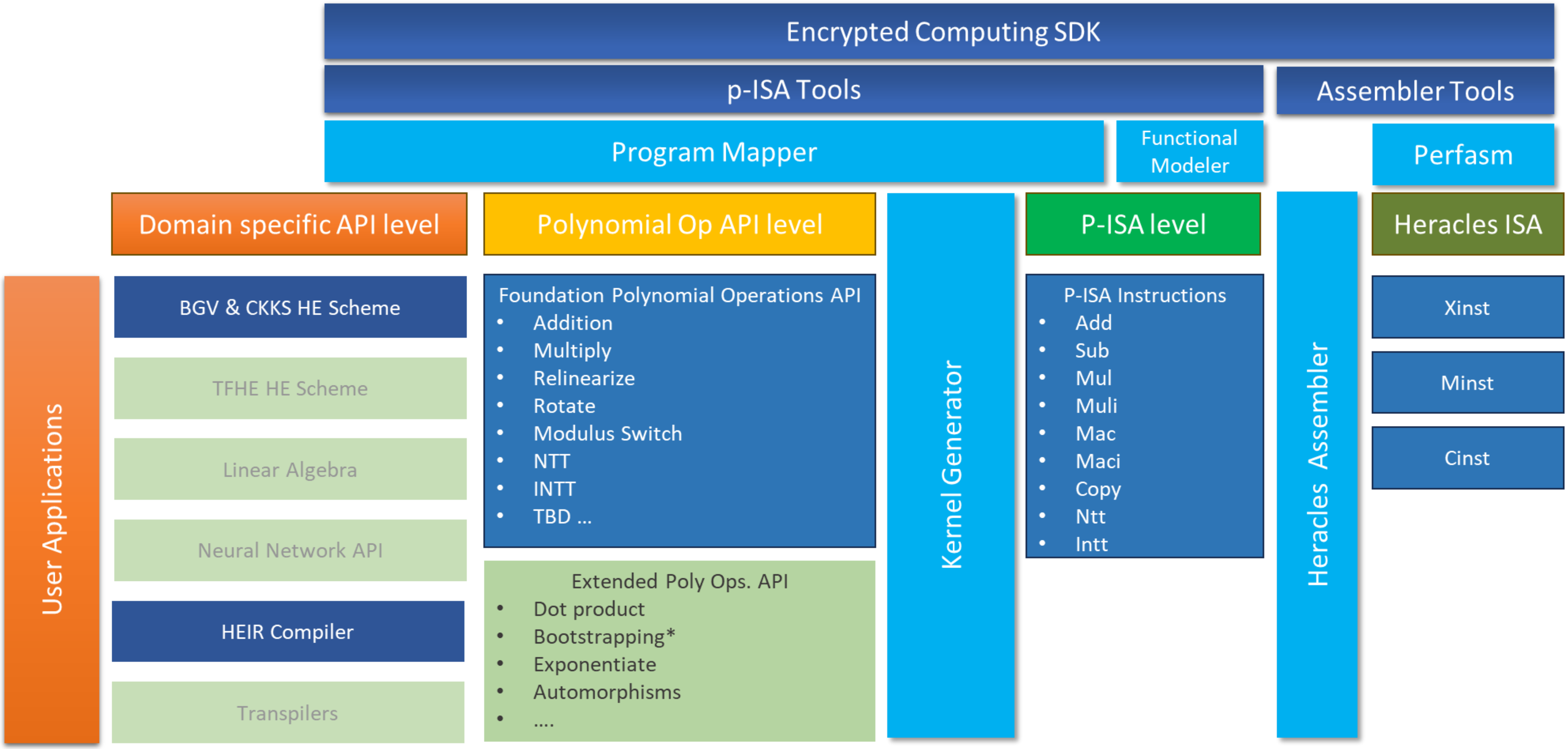
App (e.g., C++, Python, etc)

HEIR Compiler

HERACLES Overview


Distributed memory hierarchy
CXL I/O
Memory physical organization
HBM
SPAD
SPAD
HBM
Compute Engine
w tightly connected functional units
Mfetch Inst.
Cfetch Inst.
Xecute Inst

Encrypted Computing SDK Components Overview



Encrypted Computing SDK

p-ISA Tools

Assembler Tools

Program Mapper

Functional Modeler

Perfasm

User Applications

Domain specific API level

BGV & CKKS HE Scheme

TFHE HE Scheme

Linear Algebra

Neural Network API

HEIR Compiler

Transpilers

Polynomial Op API level

Foundation Polynomial Operations API

- Addition
- Multiply
- Relinearize
- Rotate
- Modulus Switch
- NTT
- INTT
- TBD …

Extended Poly Ops. API

- Dot product
- Bootstrapping*
- Exponentiate
- Automorphisms
- ….

Kernel Generator

P-ISA level

P-ISA Instructions

- Add
- Sub
- Mul
- Muli
- Mac
- Maci
- Copy
- Ntt
- Intt

Heracles Assembler

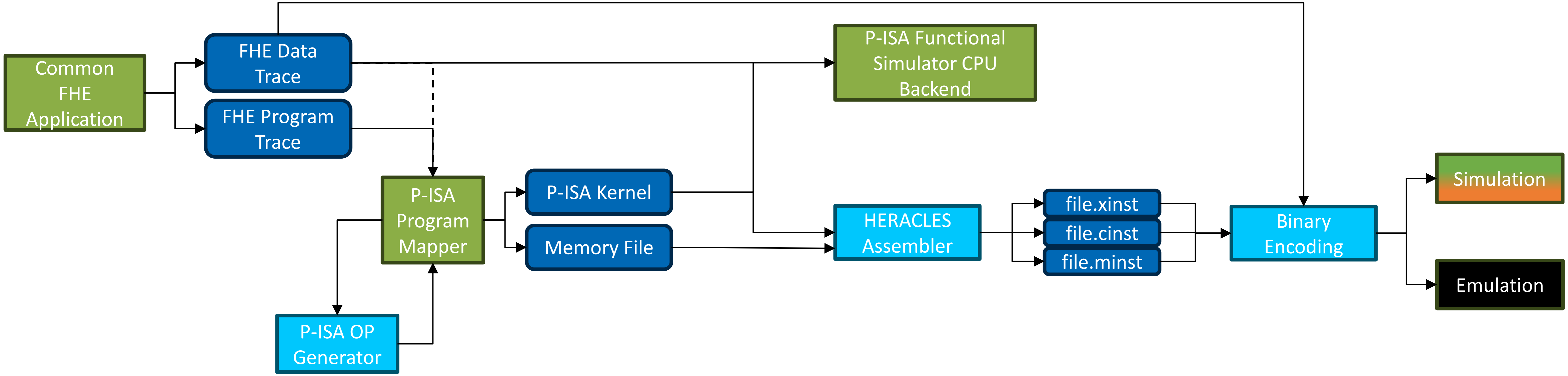
Heracles ISA

Xinst

Minst

Cinst

Current E2E Software Stack – Building Toward an Encrypted Computing SDK



Current syntax & semantic for the program trace

instruction,scheme,ring dimension,krns,arg0,arg1,arg2,…

mul_plain,BGV,16384,5,ctprod0-2-4,ct0-2-4,pt0-1-4

Instruction:Multiply Plain, *Scheme:BGV*, *Ring Dimension:16384*, *krns:5*

arg0: ***output cyphertext*** (ctprod0) with ring dimension 16384, krns 5, order 2, current rns terms 4

arg1: ***input cyphertext*** (ct0) with ring dimension 16384, krns 5, order 2, current rns terms 4

arg2: ***input cyphertext*** (ct1) with ring dimension 16384, krns 5, order 1, current rns terms 4

Revised/proposed syntax & semantic for the program trace

Scheme, Ring Dimension, Instruction, arg1 Ctxt(+params), arg2 Ctxt(+params),...
Ctxt(+params) = Ctxt label, RNS Primes, Order, Mult. Depth, Level

CKKS,16384,MULT,ct01f9,8,3,2,2,ct018d,8,2,1,2,ct01ba,8,2,1,2,

Scheme: CKKS, *Ring Dimension: 16384,* *Operation: Multiplication*

Input cyphertexts (ct018d and ct01ba) each with ring dimension 16384, 8 RNS primes, order 2, depth 1, and level2

Output cyphertext (ct01f9) with ring dimension 16384, 8 RNS primes, order 3, depth 2, and level 2

CKKS,16384,RELIN,ct0298,8,2,2,2,ct01f9,8,3,2,2,

Input cyphertexts (ct01f9) with ring dimension 16384, 8 RNS primes, order 3, depth 2, and level 2

Output cyphertext (ct0298) with ring dimension 16384, 8 RNS primes, order 2, depth 2, and level 2

Questions ?