FHE for all

Practical Applications on Hardware Platforms with HEIR

Shruthi Gorantala
Software Engineer, Google

Wouter Legiest
Student Researcher, Google
Ph.D. Student, COSIC

Alex, Asra, Jeremy and many collaborators

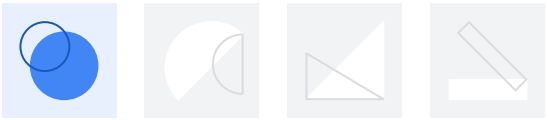
Agenda

Real World FHE

HEIR - What's New

Hardware Accelerator Demos

What's Next



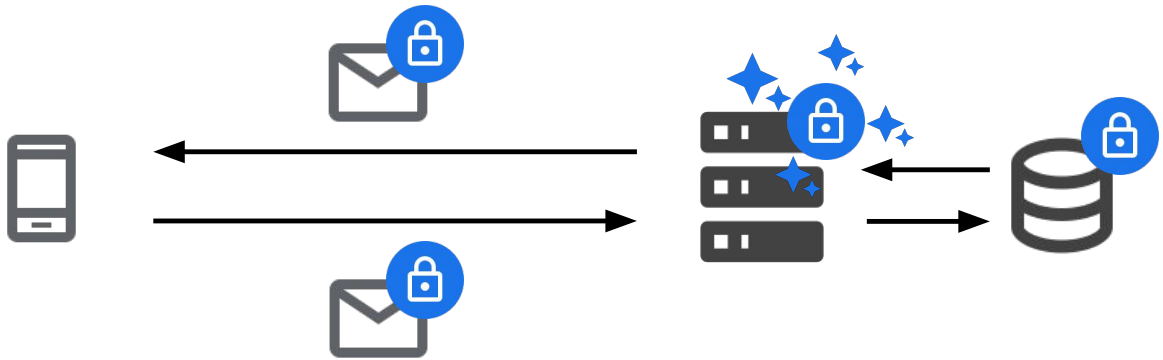
Real World FHE

Using FHE for private data processing

Google

Security and Privacy Research

FHE enables privacy at compute time



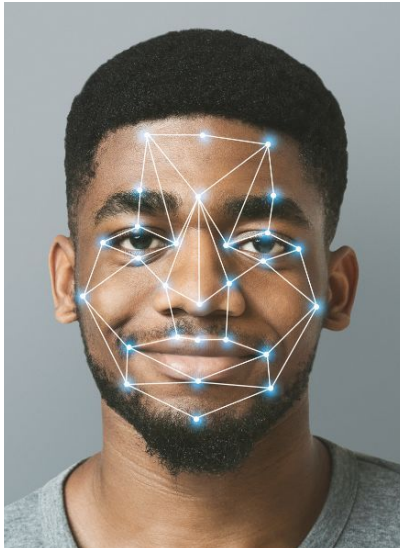
What problems does FHE solve?

Insider risk

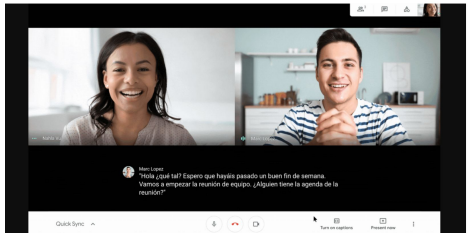
Policy compliance

Data minimization

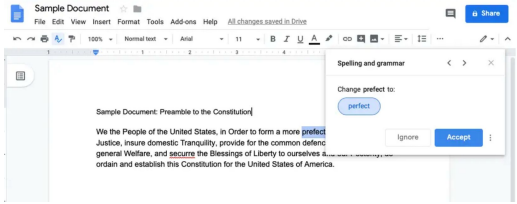
Real World Applications

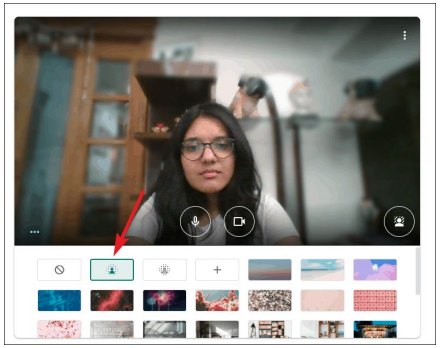


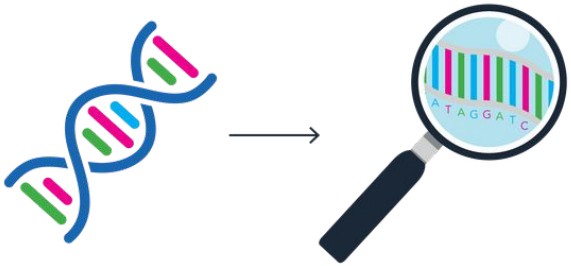
Video captions



Spell Check







Gemini

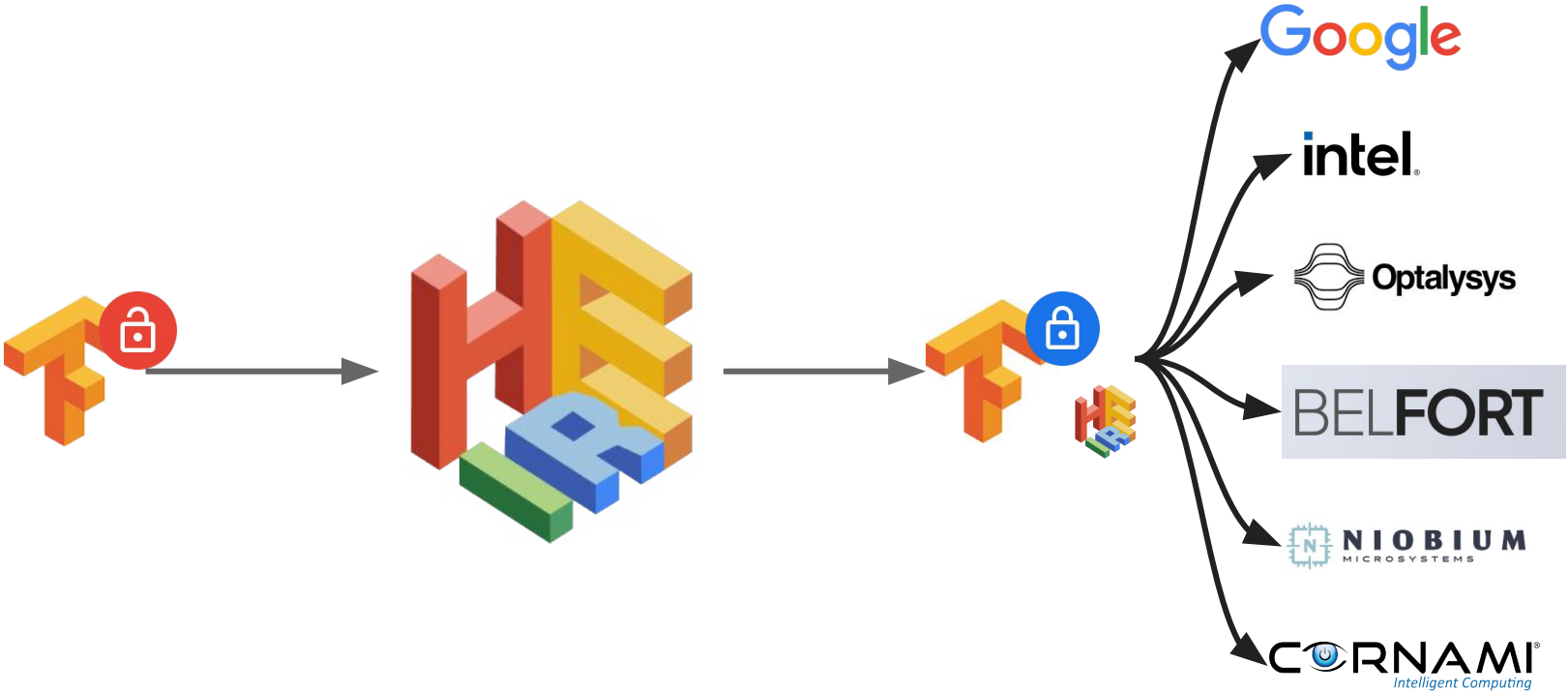
Challenges

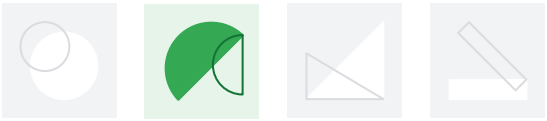
Performance

Usability

Cryptography Expertise

HEIR for FHE and privacy





HEIR

What’s new?

Google

Security and Privacy Research

Compiler for Fully Homomorphic Encryption

We're building a compiler toolchain

with established ideas in

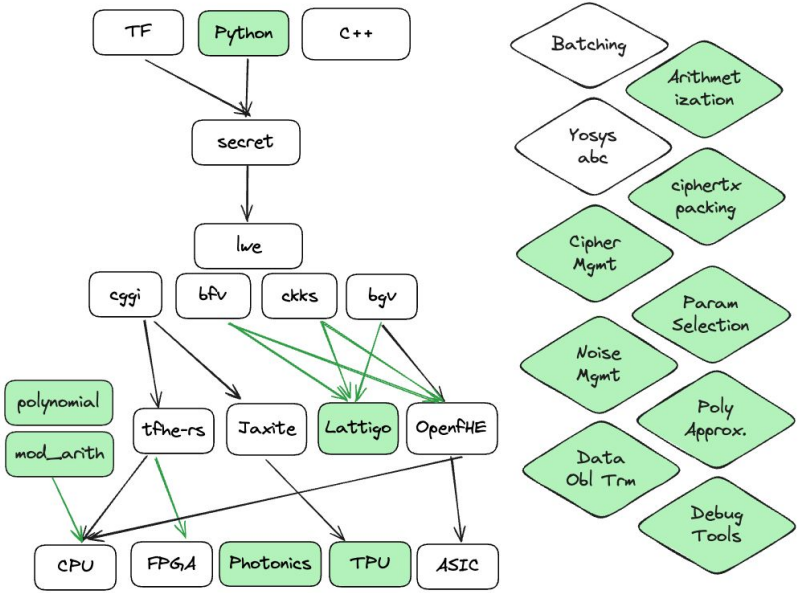
Fully Homomorphic Encryption

to accelerate research

and lower the barrier to production

Homomorphic **E**ncryption **I**ntermediate **R**epresentation

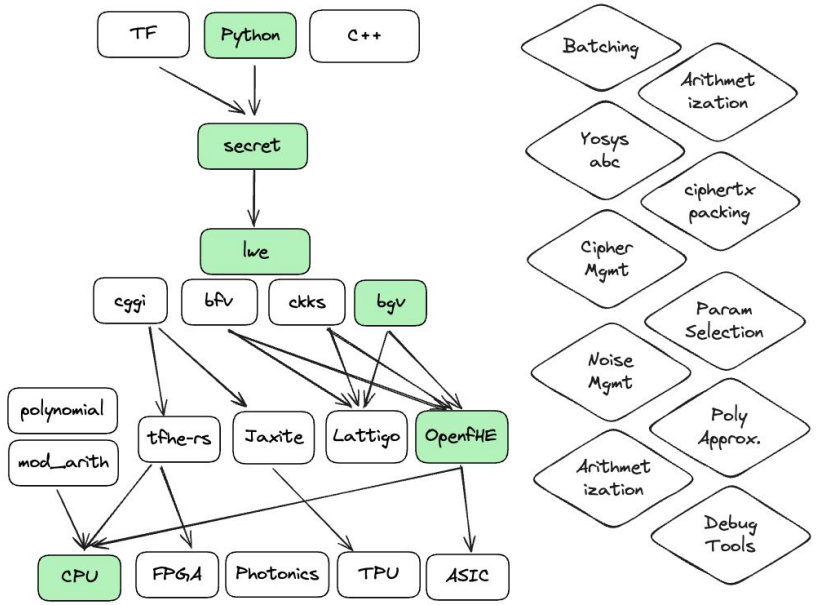
FHE Design Environment



✨ #1 requested feature ✨

Python Frontend

```python
from heir import compile
from heir.mlir import I16, Secret

# defaults to scheme="bgv", OpenFHE backend, and debug=False
@compile()
def func(x: Secret[I16], y: Secret[I16]):
  sum = x + y
  diff = x - y
  mul = x * y
  expression = sum * diff + mul
  deadcode = expression * mul
  return expression
```

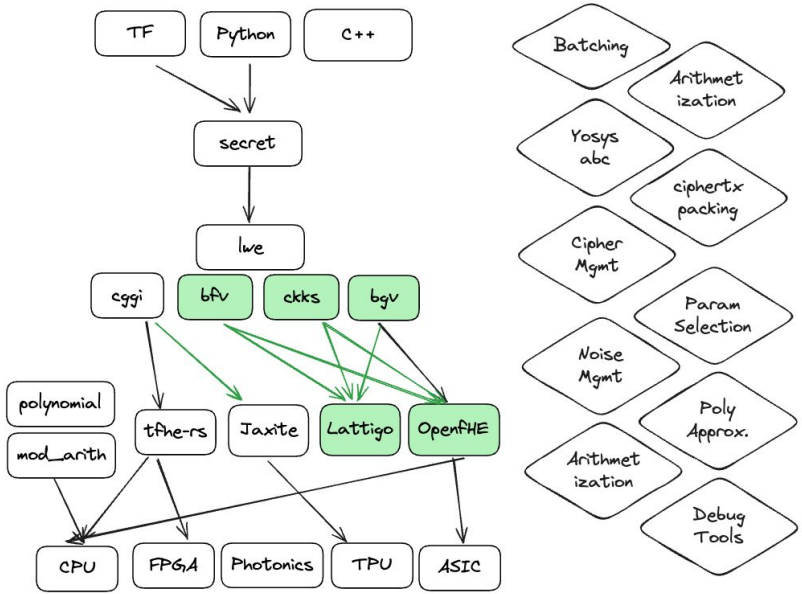


Scheme & Library support

- Last year:
  - CGGI (tfhe-rs)
  - BGV (OpenFHE)
- New:
  - BGV (OpenFHE + Lattigo)
  - BFV (OpenFHE + Lattigo)
  - CKKS (OpenFHE + Lattigo)
  - CGGI (Jaxite)

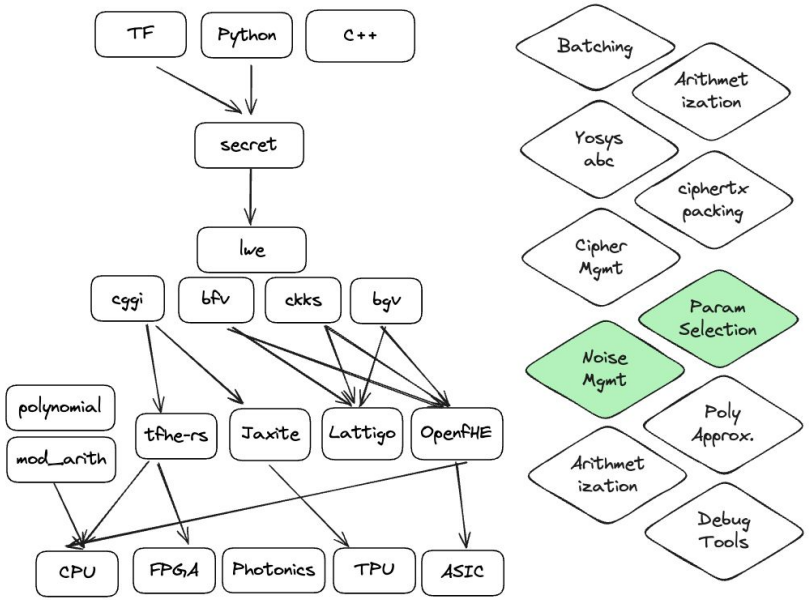
Planned:

- CKKS (Jaxite)

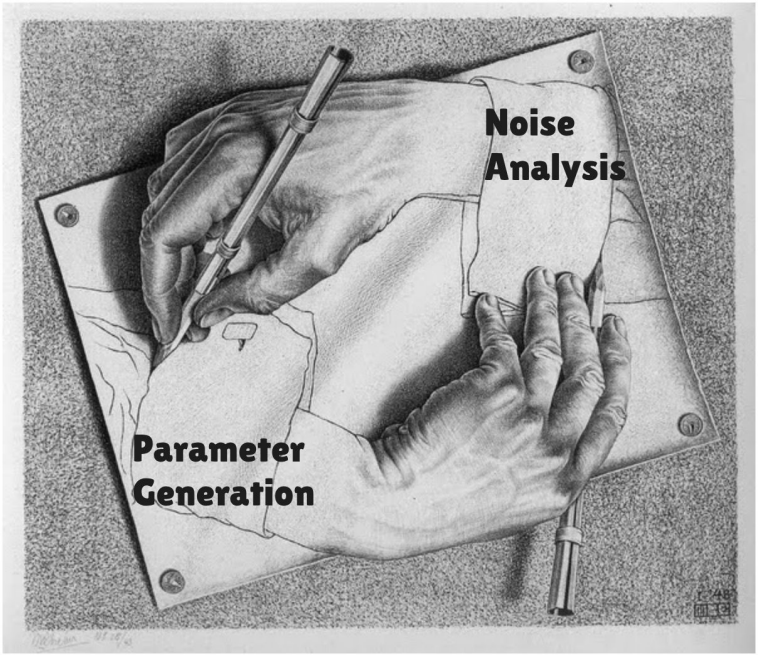


Noise modeling & parameter selection

```
func.func @dot_product(
    %ct: !ct_L2 {noise.bound = "33.47"},
    %ct_0: !ct_L2 {noise.bound = "33.47"}
) -> !ct_L0 {
  %c1 = arith.constant 1 : index
  %c2 = arith.constant 2 : index
  %c4 = arith.constant 4 : index
  %ct_1 = bgv.mul %ct, %ct_0 {noise.bound = "66.94"}
  %ct_2 = bgv.relinearize %ct_1 {
    from_basis = array<i32: 0, 1, 2>,
    to_basis = array<i32: 0, 1>,
    noise.bound = "66.94"}
  %ct_3 = bgv.rotate_cols %ct_2 {noise.bound = "66.94", offset = 4}
  %ct_4 = bgv.add %ct_2, %ct_3 {noise.bound = "67.94"}
  %ct_5 = bgv.rotate_cols %ct_4 {noise.bound = "67.94", offset = 2}
  %ct_6 = bgv.add %ct_4, %ct_5 {noise.bound = "68.94"}
  %ct_7 = bgv.rotate_cols %ct_6 {noise.bound = "68.94", offset = 1}
  %ct_8 = bgv.add %ct_6, %ct_7 {noise.bound = "69.94"}
  %ct_9 = bgv.modulus_switch %ct_8 {noise.bound = "29.73",
                                    to_ring = #ring_rns_L1_1_x8}
  %ct_10 = bgv.extract %ct_9, %c7 {noise.bound = "50.43"}
  %ct_11 = bgv.modulus_switch %ct_10 {noise.bound = "29.81",
                                      to_ring = #ring_rns_L0_1_x8}
  return %ct_11
}
```



Noise modeling & parameter selection



Noise modeling & parameter selection

Select baseline default parameters

Annotate module with scheme-specific noise models

Select specific parameters depending on target

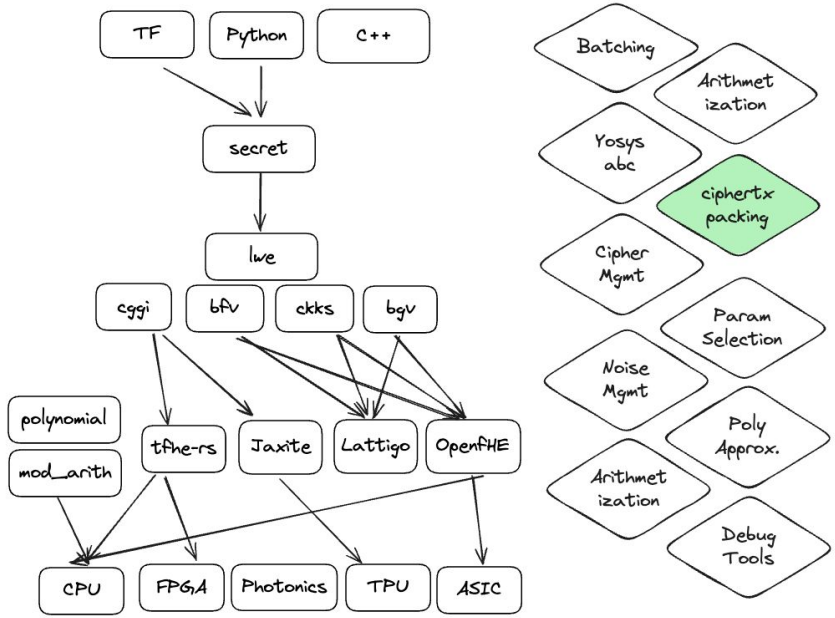
```
func.func @multiplication(
    %ct: !ct_L2 {noise.bound = "33.47"},
    %ct_0: !ct_L2 {noise.bound = "33.47"}
) -> !ct_L0 {
  %ct_1 = bgv.mul %ct, %ct_0 {noise.bound = "66.94"}
  return %ct_1
}
```

Ciphertext packing framework

- Packing notation
  - Generalizes TileTensors & Fhelipe notations.
- Fhelipe-like packing heuristic
- Halevi-Shoup-style diagonal packings
- Vos-Vos-Erkin 22 shift networks

Planned:

- More linear algebra kernels
- Integration with ML frontends
- New packing optimizers



Layout Assignment

1

Initial Layout Assignment

- Assign row-major layout for all inputs and plaintext constants
- Assign conversions and default kernels for all operations

2

Layout Optimization

- Fhelipe-like heuristic (bias towards early layout conversions)
- Annotate kernels (e.g. Halevi-Shoup)

3

Realize Ciphertext Semantics

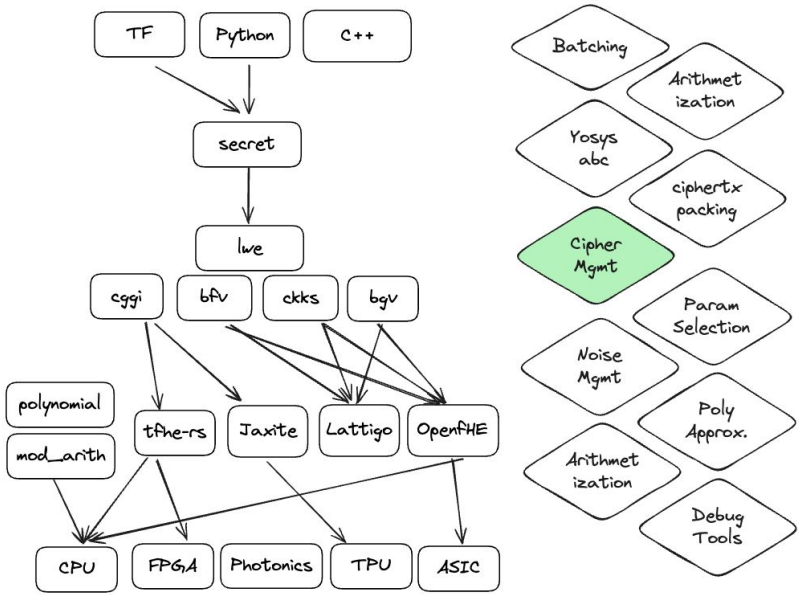
- Convert to resulting ciphertext types
- Realize kernels as FHE operations and layout conversions with shift networks

Ciphertext management

- Local & global optimization passes
- Global relinearization placement (ILP)
- Baseline methods for bootstrap, modulus switch placement, etc.

Planned:

- Scale management
- Porting more existing techniques

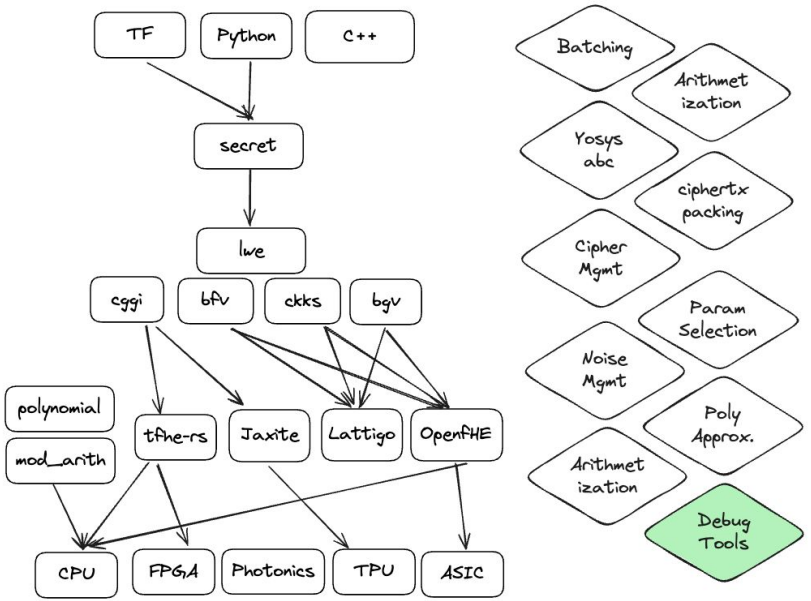


Codegen debug handler

```
--scheme-to-lattigo=
  insert-debug-handler-calls=true
```

Decrypt and inspect the ciphertext at each step!

```
Input
  Noise: 6.34 Total: 80
  Noise Bound: 12.77 Gap: 6.43

Input
  Noise: 6.38 Total: 80
  Noise Bound: 12.77 Gap: 6.39

lattigo.bgv.mul
  Noise: 34.13 Total: 80
  Noise Bound: 43.80 Gap: 9.67

lattigo.bgv.relinearize
  Noise: 34.13 Total: 80
  Noise Bound: 43.80 Gap: 9.67

lattigo.bgv.rotate_columns
  Noise: 34.67 Total: 80
  Noise Bound: 44.80 Gap: 10.13

...
```

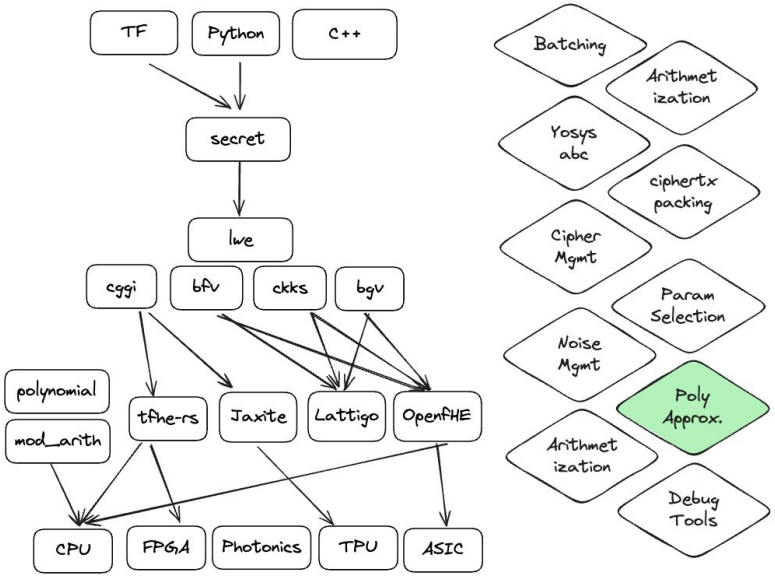


Polynomial Approximation

- Compile-time approximation solver based on Caratheodory-Fejer
- General polynomial evaluation IR
- Lowering via Paterson-Stockmeyer

Planned:

- Chebyshev basis support
- More evaluation methods (BSGS, Vos-Conti-Erkin 24)
- Integration with ML frontends

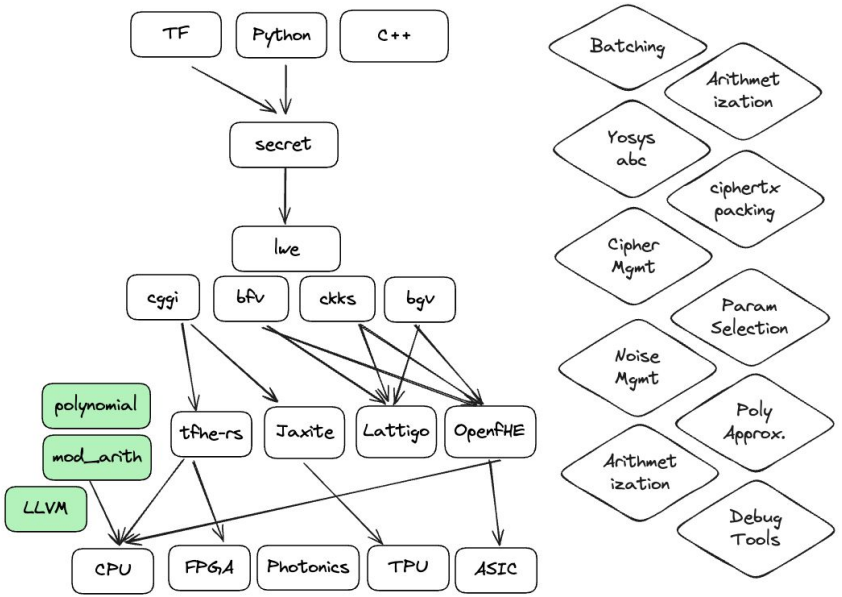


Low-level dialects

- `polynomial` dialect
  - ntt/intt ops and lowering to CPU
- `mod_arith` dialect for precise modular arithmetic semantics
- `rns` dialect for representing residue number system types
- Lowerings to LLVM

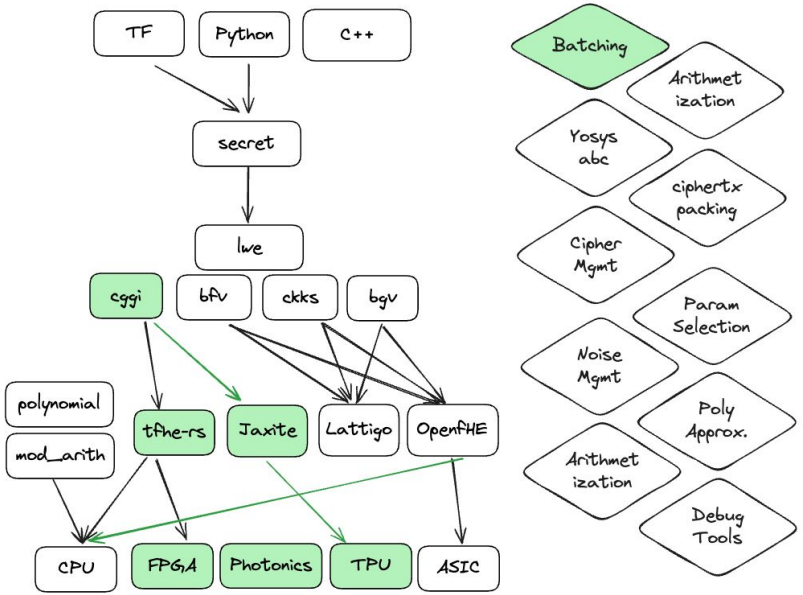
Planned:

- Hardware exit dialects (e.g., polynomial ISA)
- Lowering FHE scheme dialects to polynomial
- Low-level canonicalization patterns



Hardware

- Integration with Belfort/FPT FPGA
  - tfhe-rs boolean API intermediary
- Integration with Optalysys hardware
  - Optalysys C API intermediary
- Integration with TPU
  - Jaxite API



Arithmetization for RLWE Schemes

```
math.exp %x {
        degree = 3 : i32,
        domain_lower = -1.0 : f64,
        domain_upper = 1.0 : f64
} : f32
```

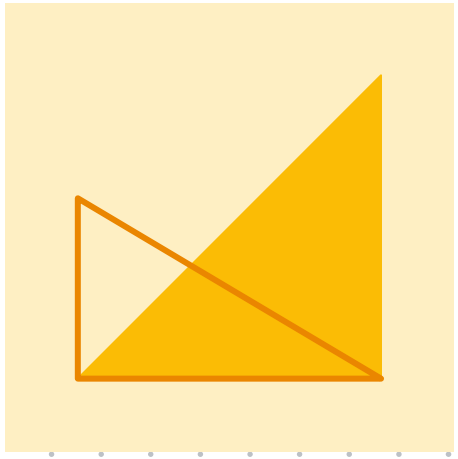
Non-linear functions (exp, abs, max, etc)

```
%0 = polynomial.eval #polynomial<
   0.99457947632469512 + 0.9956677100276301x
 + 0.5429727883818608x**2 + 0.17953348361617388x**3>,
  %x : f32
```

Polynomial approximation

HEIR for hardware

FHE on hardware accelerators

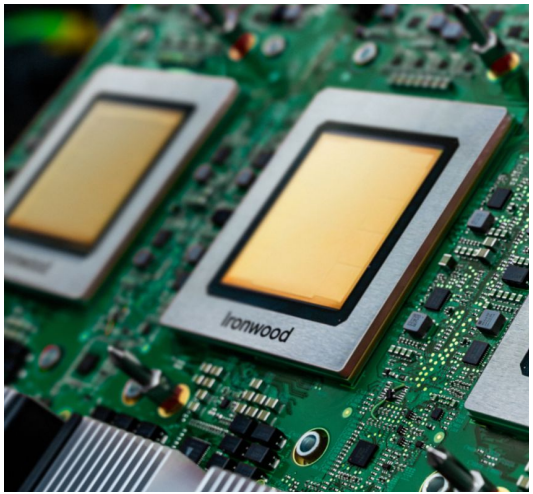


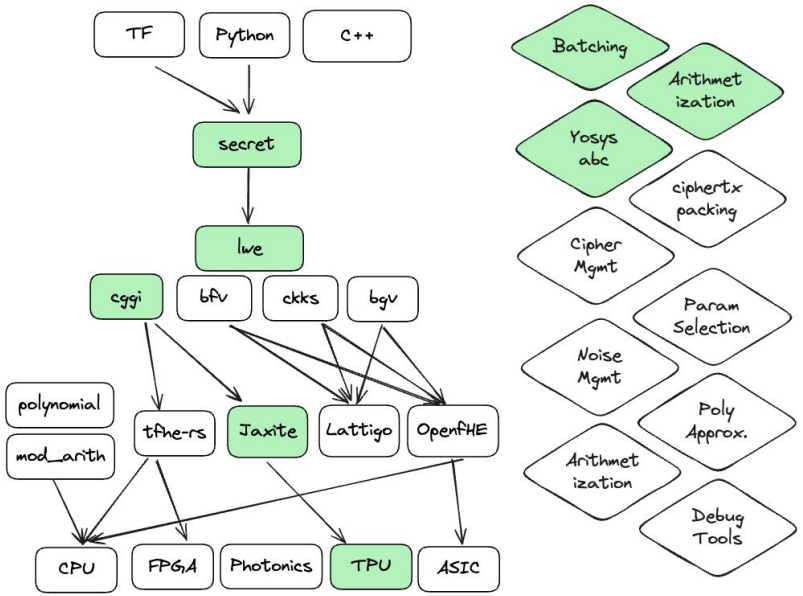
Google

Security and Privacy Research

Demo Time

CGGI HW - TPU


Ironwood


TF
Python
C++
secret
lwe
cggi
bfv
ckks
bgv
polynomial
mod_arith
tfhe-rs
Jaxite
Lattigo
OpenFHE
CPU
FPGA
Photonics
TPU
ASIC
Batching
Arithmetization
Yosys abc
ciphertx packing
Cipher Mgmt
Param Selection
Noise Mgmt
Poly Approx.
Arithmetization
Debug Tools

CGGI HW - TPU

- And gate (8ms)
- Add-one
  - **Sequential (1 TPU device)**
  - Parallel (8 TPU devices)

heir-opt

--mlir-to-cggi=abc-fast=true

--scheme-to-jaxite="parallelism=1"

heir-translate

--emit-jaxite

```
def main(v0: list[types.LweCiphertext], v1: jaxite_bool.ServerKeySet, v2: jaxite_bool.Parameters) ->
list[types.LweCiphertext]:
 temp_nodes: Dict[int, Any] = {}
 temp_nodes[19] = v0[1]
 temp_nodes[20] = v0[0]
 temp_nodes[21] = jaxite_bool.constant(False, v2)
 temp_nodes[22] = jaxite_bool.lut3(temp_nodes[20], temp_nodes[19], temp_nodes[21], 6, v1, v2)
 temp_nodes[23] = v0[2]
 temp_nodes[24] = jaxite_bool.lut3(temp_nodes[20], temp_nodes[19], temp_nodes[23], 120, v1, v2)
 temp_nodes[25] = jaxite_bool.lut3(temp_nodes[20], temp_nodes[19], temp_nodes[23], 128, v1, v2)
 temp_nodes[26] = v0[3]
 temp_nodes[27] = jaxite_bool.lut3(temp_nodes[25], temp_nodes[26], temp_nodes[21], 6, v1, v2)
 temp_nodes[28] = v0[4]
 temp_nodes[29] = jaxite_bool.lut3(temp_nodes[25], temp_nodes[26], temp_nodes[28], 120, v1, v2)
 temp_nodes[30] = jaxite_bool.lut3(temp_nodes[25], temp_nodes[26], temp_nodes[28], 128, v1, v2)
 temp_nodes[31] = v0[5]
 temp_nodes[32] = jaxite_bool.lut3(temp_nodes[30], temp_nodes[31], temp_nodes[21], 6, v1, v2)
 temp_nodes[33] = v0[6]
 temp_nodes[34] = jaxite_bool.lut3(temp_nodes[30], temp_nodes[31], temp_nodes[33], 120, v1, v2)
 temp_nodes[35] = jaxite_bool.lut3(temp_nodes[30], temp_nodes[31], temp_nodes[33], 128, v1, v2)
 temp_nodes[36] = v0[7]
 temp_nodes[37] = jaxite_bool.lut3(temp_nodes[35], temp_nodes[36], temp_nodes[21], 6, v1, v2)
 temp_nodes[38] = jaxite_bool.lut3(temp_nodes[20], temp_nodes[21], temp_nodes[21], 1, v1, v2)
 temp_nodes[39] = np.full((8), None)
           return temp_nodes[39]
```

```
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$ python3 jaxite_example.py
```

CGGI HW - TPU

- And gate (8ms)
- Add-one
  - Sequential (1 TPU device)
  - **Parallel (8 TPU devices)**

heir-opt

--mlir-to-cggi=abc-fast=true

--scheme-to-jaxite="**parallelism=8**"

heir-translate

--emit-jaxite

```
def test_add_one_lut3(
 v0: list[types.LweCiphertext],
 v1: jaxite_bool.ServerKeySet,
 v2: jaxite_bool.Parameters,
) -> list[types.LweCiphertext]:
 temp_nodes: Dict[int, Any] = {}
 temp_nodes[19] = v0[1]
 temp_nodes[20] = v0[0]
 temp_nodes[21] = jaxite_bool.constant(False, v2)
 temp_nodes[23] = jaxite_bool.lut3(temp_nodes[20], temp_nodes[19], temp_nodes[22], 128, v1, v2)
 temp_nodes[26] = jaxite_bool.lut3(temp_nodes[23], temp_nodes[24], temp_nodes[25], 128, v1, v2)
 temp_nodes[29] = jaxite_bool.lut3(temp_nodes[26], temp_nodes[27], temp_nodes[28], 128, v1, v2)
 inputs = [(temp_nodes[21], temp_nodes[27], temp_nodes[26], 6),
           (temp_nodes[28], temp_nodes[27], temp_nodes[26], 120),
           (temp_nodes[21], temp_nodes[24], temp_nodes[23], 6),
           (temp_nodes[21], temp_nodes[19], temp_nodes[20], 6),
           (temp_nodes[25], temp_nodes[24], temp_nodes[23], 120),
           (temp_nodes[21], temp_nodes[21], temp_nodes[20], 1),
           (temp_nodes[22], temp_nodes[19], temp_nodes[20], 120),
           (temp_nodes[21], temp_nodes[30], temp_nodes[29], 6),]
 temp_nodes[40] = jaxite_bool.pmap_lut3(inputs, v1, v2)
 temp_nodes[49] = np.full((8), None)
 temp_nodes[49][0] = temp_nodes[46]
 temp_nodes[49][1] = temp_nodes[44]
 temp_nodes[49][2] = temp_nodes[47]
 temp_nodes[49][3] = temp_nodes[43]
 temp_nodes[49][4] = temp_nodes[45]
 temp_nodes[49][5] = temp_nodes[41]
 temp_nodes[49][6] = temp_nodes[42]
 temp_nodes[49][7] = temp_nodes[48]
 return temp_nodes[49]
```

```
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$ python3 add_one/add_one_main.py
```

Small ML Model on TPU

- Hello World
  - Feed forward neural network (3x3)

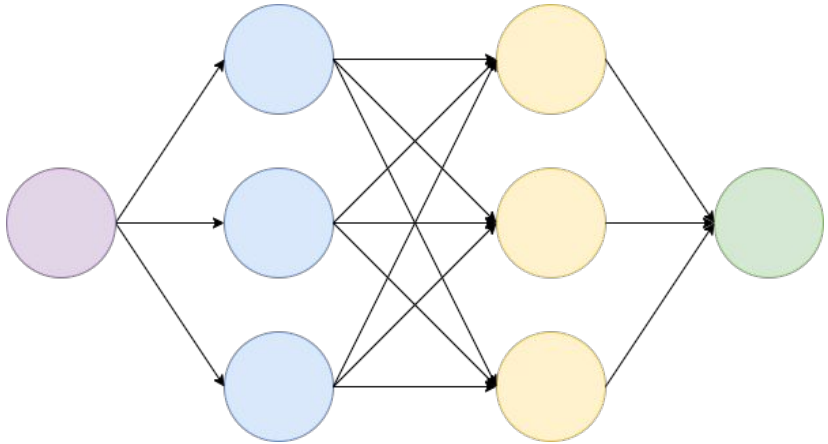
heir-opt

--mlir-to-cggi=abc-fast=true

--scheme-to-jaxite="parallelism=8"

heir-translate

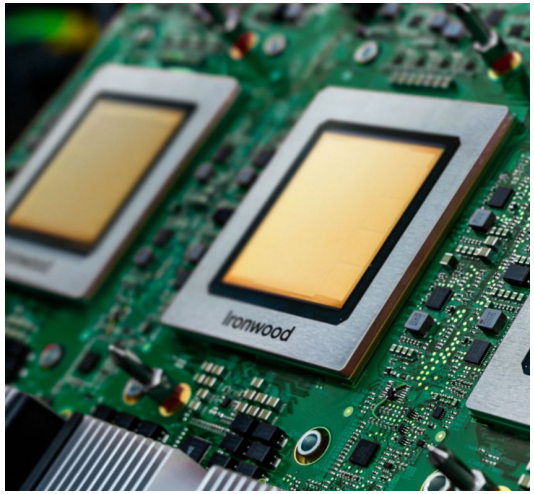
--emit-jaxite



```
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$
gshruthi_google_com@t1v-n-e5ca4fd2-w-0:~$ python3 hello_world_small/hello_world_s
mall_main.py
```

Next : HEIR - TPU

- CGGI
  - Custom kernels
  - Multi-host TPU device scheduling
- CKKS
  - Custom NTT kernels (5x over A100)
  - Enable E2E encrypted ML inference



CGGI HW - Belfort

- Integration with Belfort/FPT FPGA
  - tfhe-rs boolean API intermediary
  - Multi-FPGA support
  - PBS batching/scheduling
  - 200x speedup vs single-threaded CPU
  - Continuing work on higher-level int APIs

BELFORT

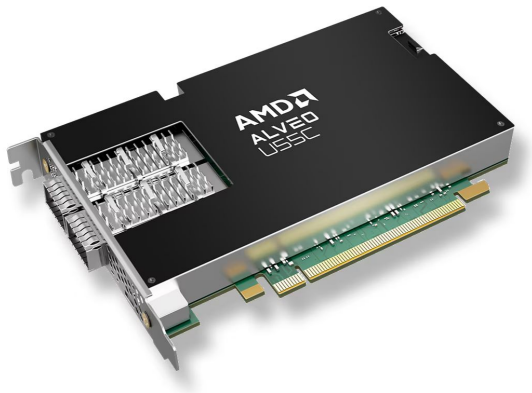
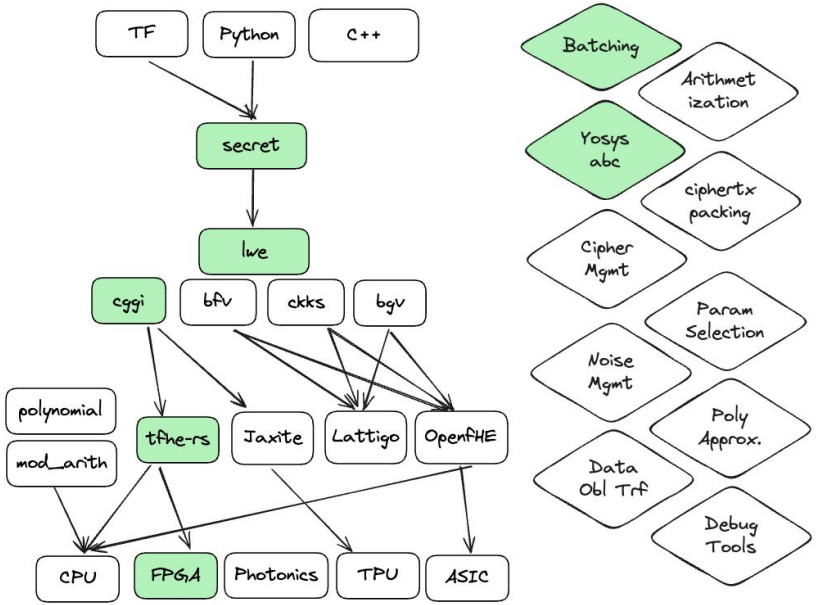


Photo from amd.com

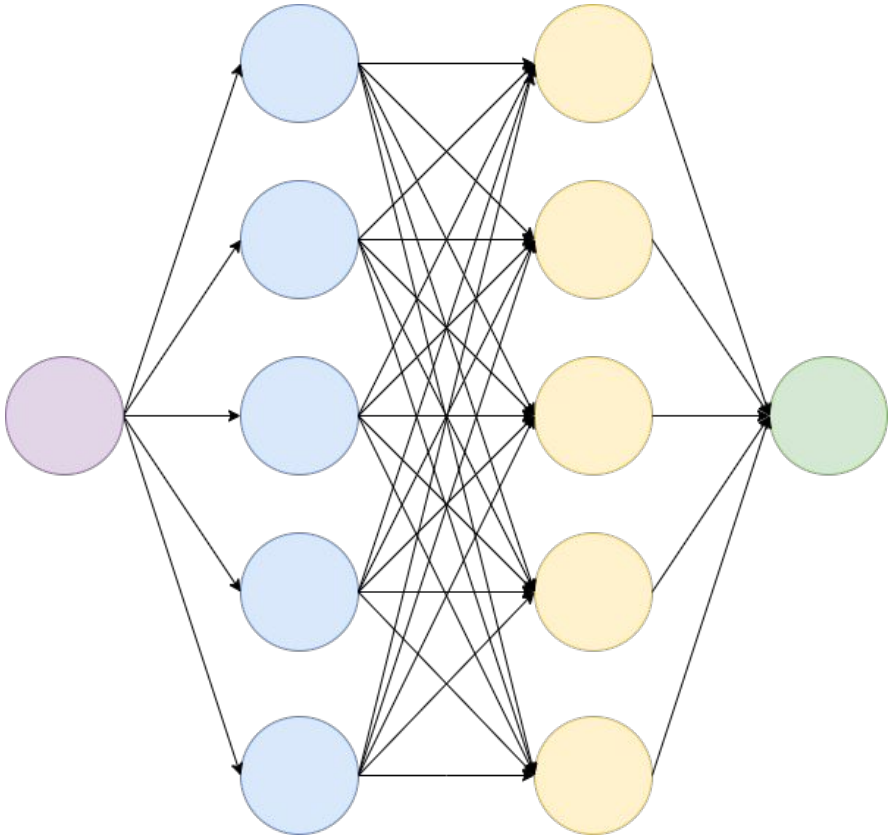
CGGI HW - Belfort

- Integration with Belfort/FPT FPGA
  - tfhe-rs boolean API intermediary
  - Multi-FPGA support
  - PBS batching/scheduling
  - 200x speedup vs single-threaded CPU
  - Continuing work on higher-level int APIs



CGGI HW - Demo

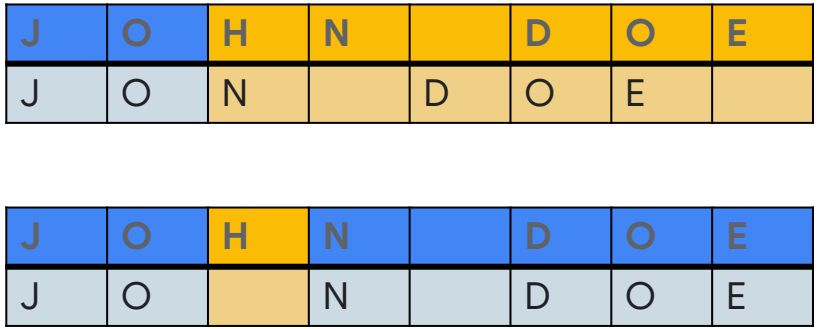
- Feed forward neural network (5x5)
  - 3 Fully connected layers with 5 nodes
  - Converted with Yosys
  - Optimized scheduling by HEIR



CGGI HW - Demo

String Comparison

- How close are two strings?
  - Natural measure
- “John Doe” vs. “Jon Doe”
  - Hamming distance: 6 difference
  - Edit distance: 1 difference

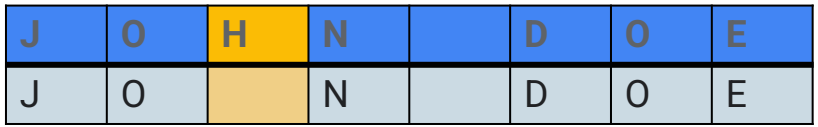


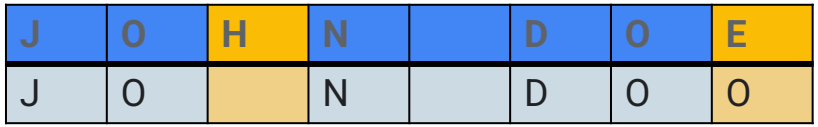
Google

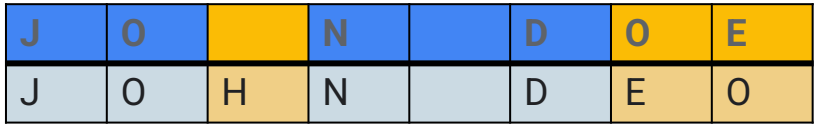
Security and Privacy Research

Edit distance

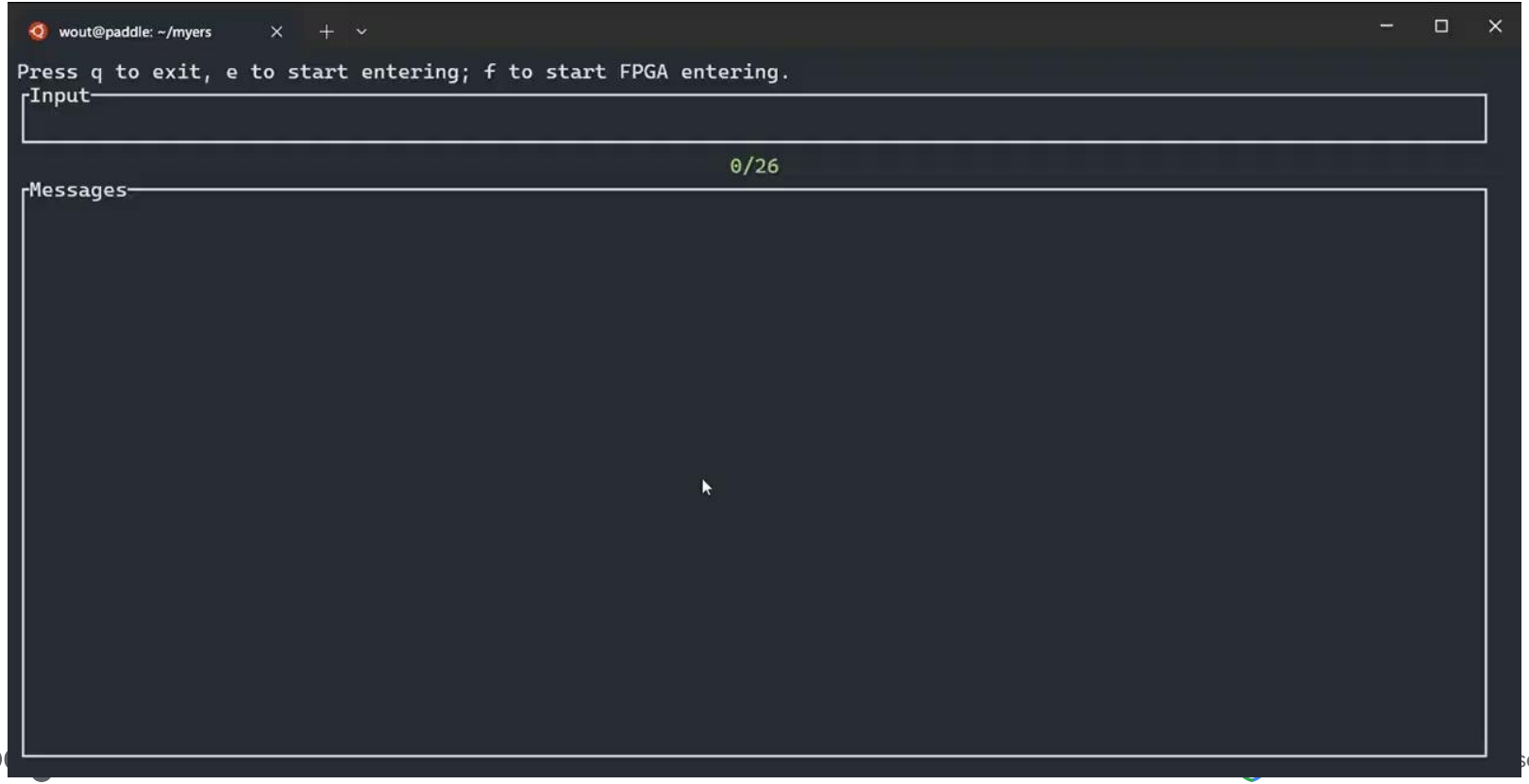
- “John Doe” vs. “Jon Doe”
  - Edit distance: 1 difference
- Minimal number of operations
  - Deletion
  - Insertion
  - Substitution



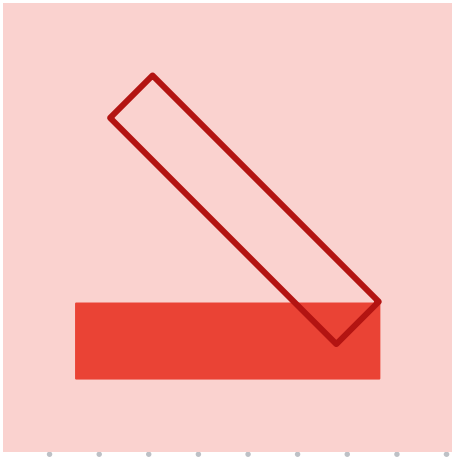




Leuvenstein



What’s next?



What's Next for HEIR

- Hardware Accelerators
- Encrypted Inference
- Support Research goals

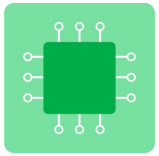
HEIR + Hardware accelerator

GPU

FPGA

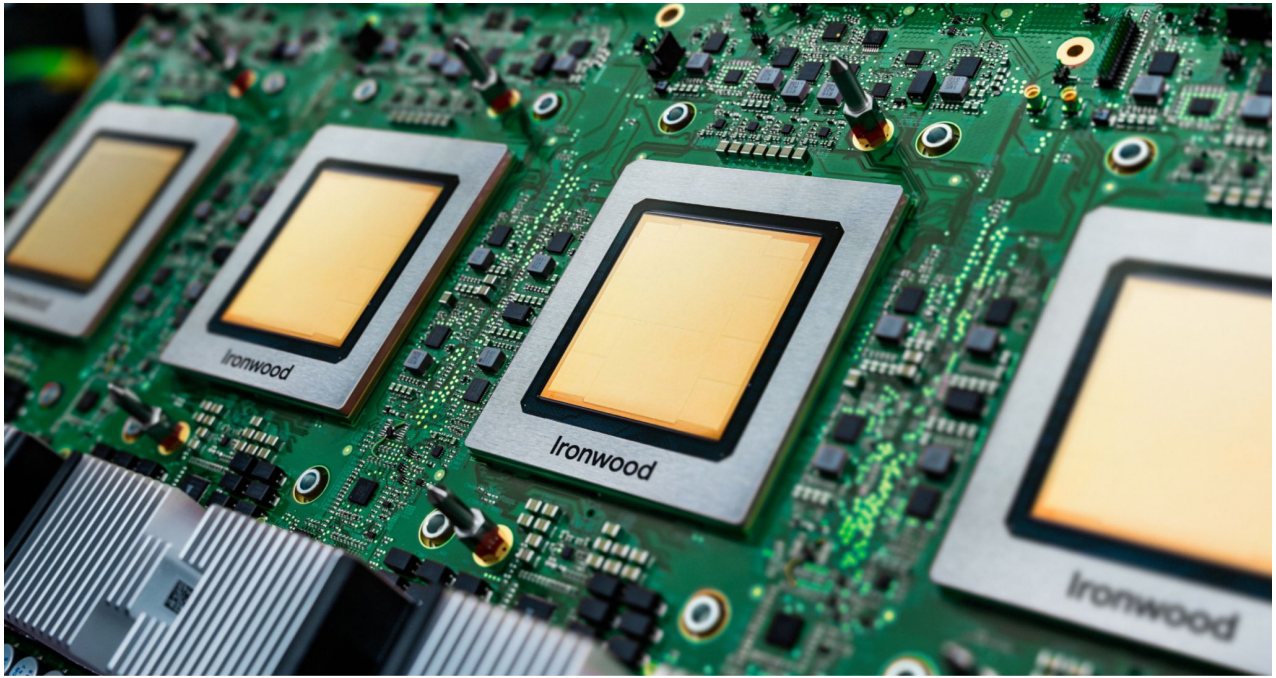
TPU

Photonics



ASIC

HEIR + Cloud TPU



HEIR + Cloud FPGA

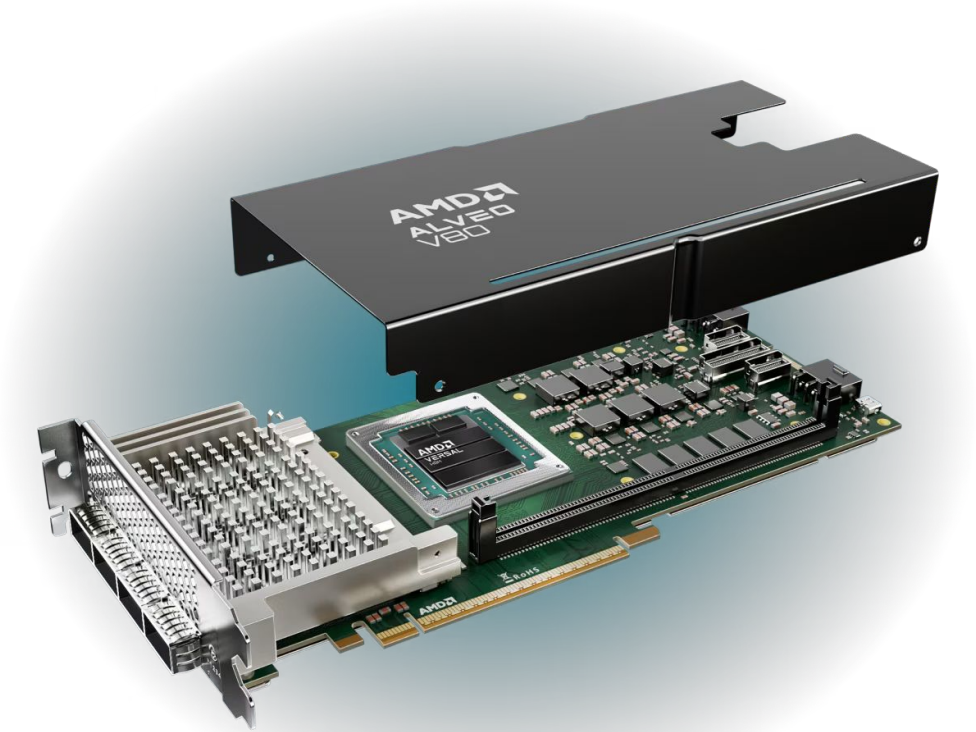
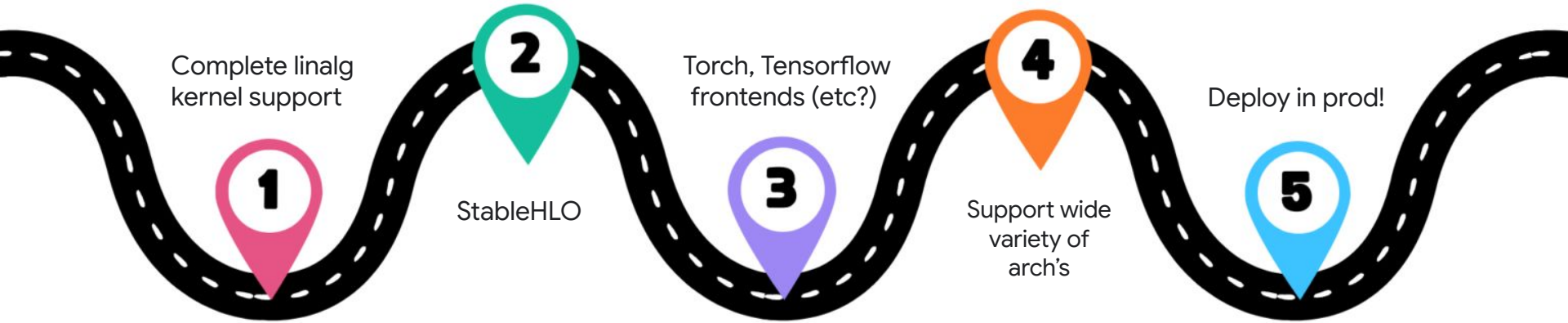


Photo from amd.com

HEIR for Standardizing FHE Hardware

Encrypted Inference



Support the Research Goals of our Community

HEIR Collaborations and Partnerships

Fully Homomorphic Encryption .ORG

BELFORT

KU LEUVEN

Exploring the transformational potential of FHE and the path toward adoption of its "stack."

BY SHRUTHI GORANTALA, ROB SPRINGER, AND BRYANT GIPSON

Unlocking the Potential of Fully Homomorphic Encryption

NIOBIUM MICROSYSTEMS

Georgia Tech

WAHC 2022 – 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography

Fraunhofer AISEC

ETH zürich

FHE Use-Cases and Benchmarking
Breakout Session

FHETCH

The industry body for Fully Homomorphic Encryption hardware

Optalysys

8th HomomorphicEncryption.org Standards Meeting

Join us!

- heir.dev
- github.com/google/heir
- #heir channel on the FHE.org discord
- Office Hours every Monday
- Open meeting once a month on Thursday

📅 Meeting calendar 📅