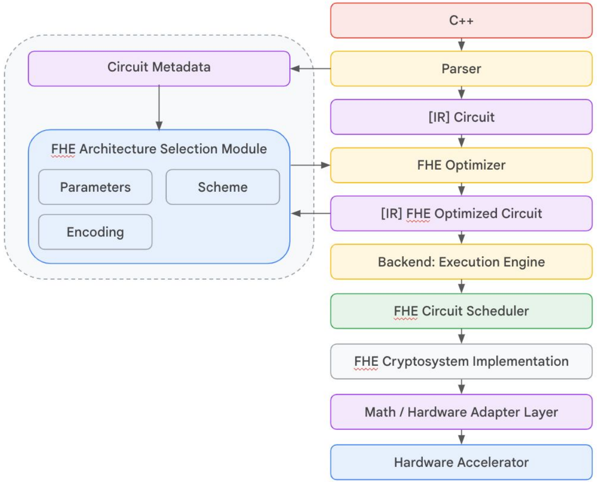
The Path to Real World FHE: Navigating the Ciphertext Space

Shruthi Gorantala
gshruthi@google.com

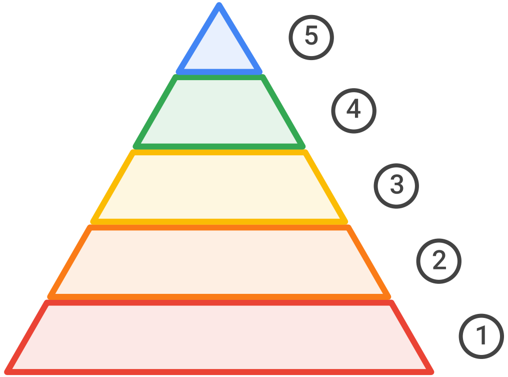
Real World Crypto
March 28, 2023

AGENDA

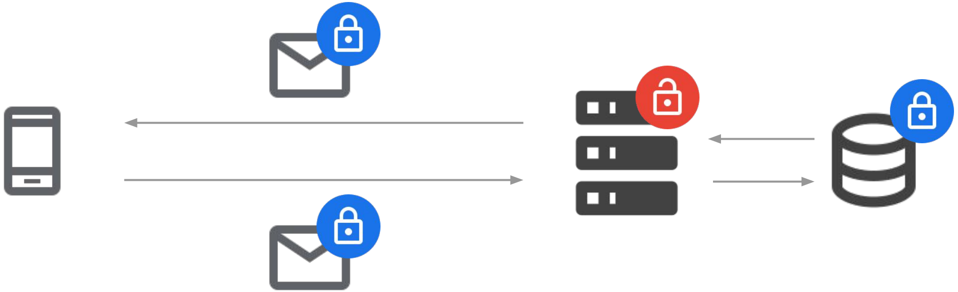
FHE Stack


C++
Circuit Metadata
Parser
[IR] Circuit
FHE Architecture Selection Module
FHE Optimizer
Parameters
Scheme
[IR] FHE Optimized Circuit
Encoding
Backend: Execution Engine
FHE Circuit Scheduler
FHE Cryptosystem Implementation
Math / Hardware Adapter Layer
Hardware Accelerator

FHE Hierarchy of Needs

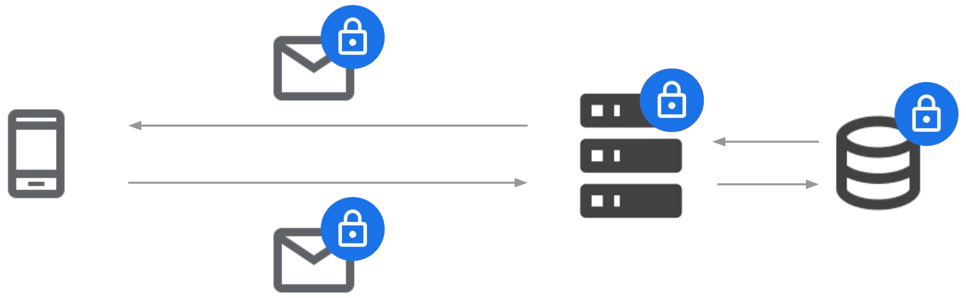


Privacy At Compute-time



Without FHE

Privacy At Compute-time



With FHE

Insider risk

Policy compliance

Data minimization

What is Fully Homomorphic Encryption

Ciphertext

$$\begin{tikzcd}
E_m \arrow[r, "FHE_f"] \arrow[d, leftarrow, "\text{Encrypt}"'] & FHE_f(E_m) \arrow[d, "\text{Decrypt}"] \\
m \arrow[r, "f"] & f(m)
\end{tikzcd}$$

Plaintext

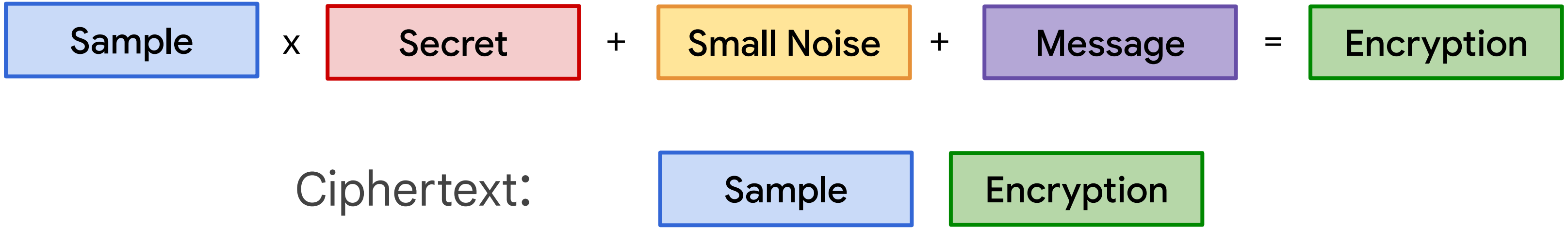
What’s that Noise?

(Ring) Learning With Errors Encryption

Sample x Secret + Small Noise + Message = Encryption

What’s that Noise?

(Ring) Learning With Errors Encryption



Sample x Secret + Small Noise + Message = Encryption

Ciphertext: Sample Encryption

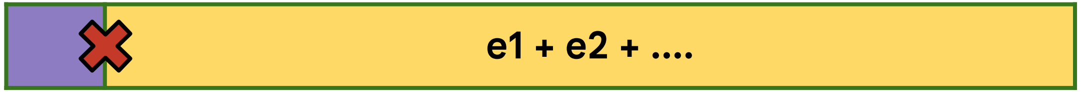
- Ciphertext sizes
  - LWE encrypts scalars: 1 B ⟶ ~20000 bits
  - RLWE encrypts vectors: 4 B ⟶ >1.7 MB

What’s that Noise?

(Ring) Learning With Errors Encryption

Sample x Secret + Small Noise + Message = Encryption

Problem: Decryption fails if error grows too large

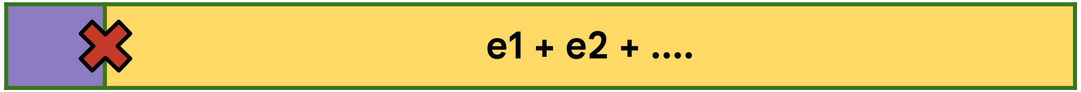


What’s that Noise?

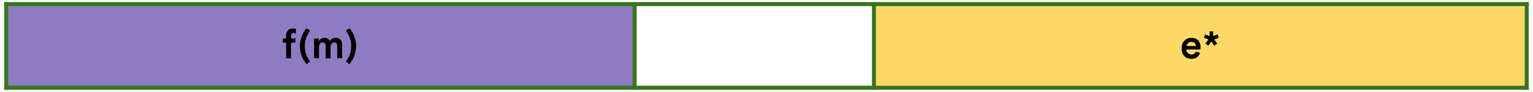
(Ring) Learning With Errors Encryption

Sample x Secret + Small Noise + Message = Encryption

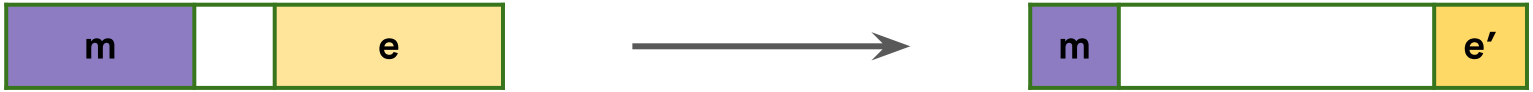
Problem: Decryption fails if error grows too large



Solution: Choose **large parameters** to fit the entire computation



Solution : Track noise and perform **ciphertext-refresh operations**



Challenges

Data size expansion

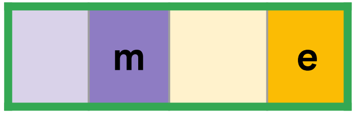
Speed

Usability

Developing in FHE

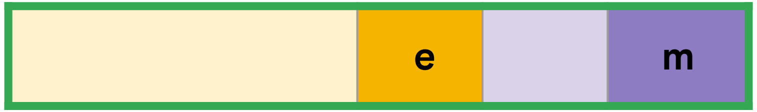
Which FHE Scheme(s) ?

Privacy, Performance & Precision



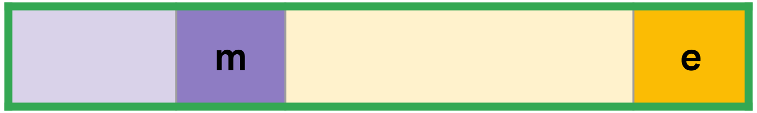
TFHE

Boolean Gates and Look up tables



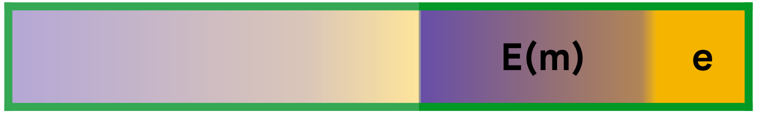
BGV

Integer Arithmetic



BFV

Integer Arithmetic



CKKS

Approximate fixed point arithmetic

Which FHE Library ?

Expose primitive operations : Gates, ADD, MUL, Ciphertext*

- TFHE
- CONCRETE - Boolean
- OpenFHE - BinFHE
- CuFHE
- NuFHE
- FHEW

- SEAL
- HEAAN
- Lattigo
- HELib
- OpenFHE - CKKS
- CONCRETE

Which FHE Compiler?

Optimizes for a specific library & few types of applications

How to interop amongst them?

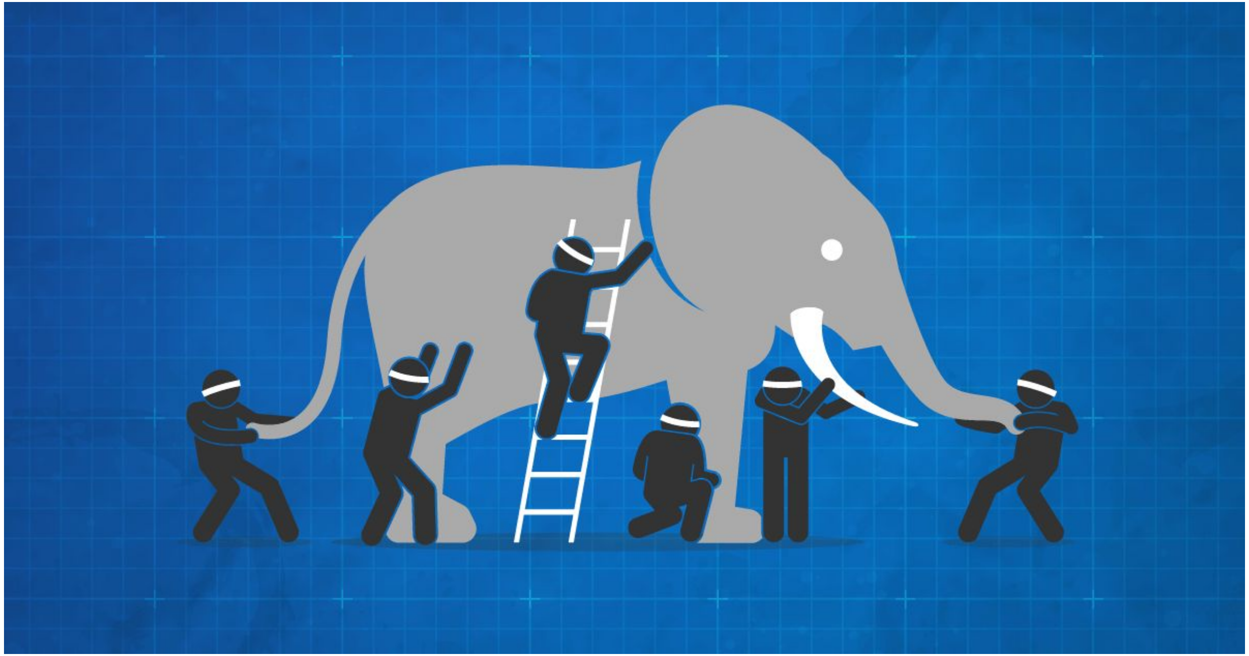
- Cingulata
- E3
- CHET
- ALCHEMY
- RAMPARTS
- MARBLE
- CONCRETE-ML
- SEAL - EVA
- nGraph-HE
- SEALion
- HELayers
- HECO

Which FHE Hardware Accelerator?

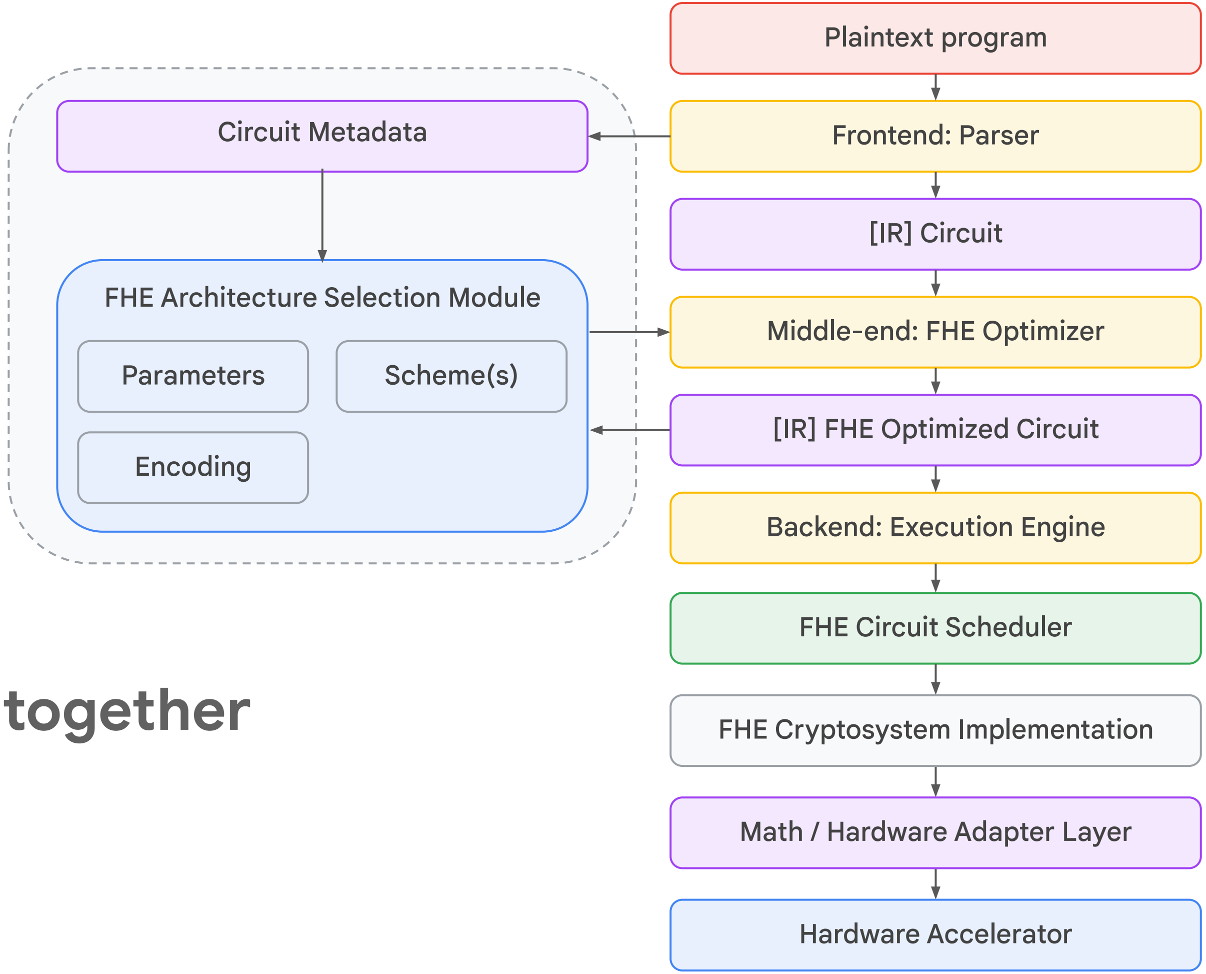
Do they exist for real?
Will they work for any application?

- Intel HEXL
- Optalysys
- F1 Accelerator
- BASALISC
- TREBUCHET
- Crater Lake
- Cornami
- DPRIVE
- FPT
- HERACLES

Research in FHE



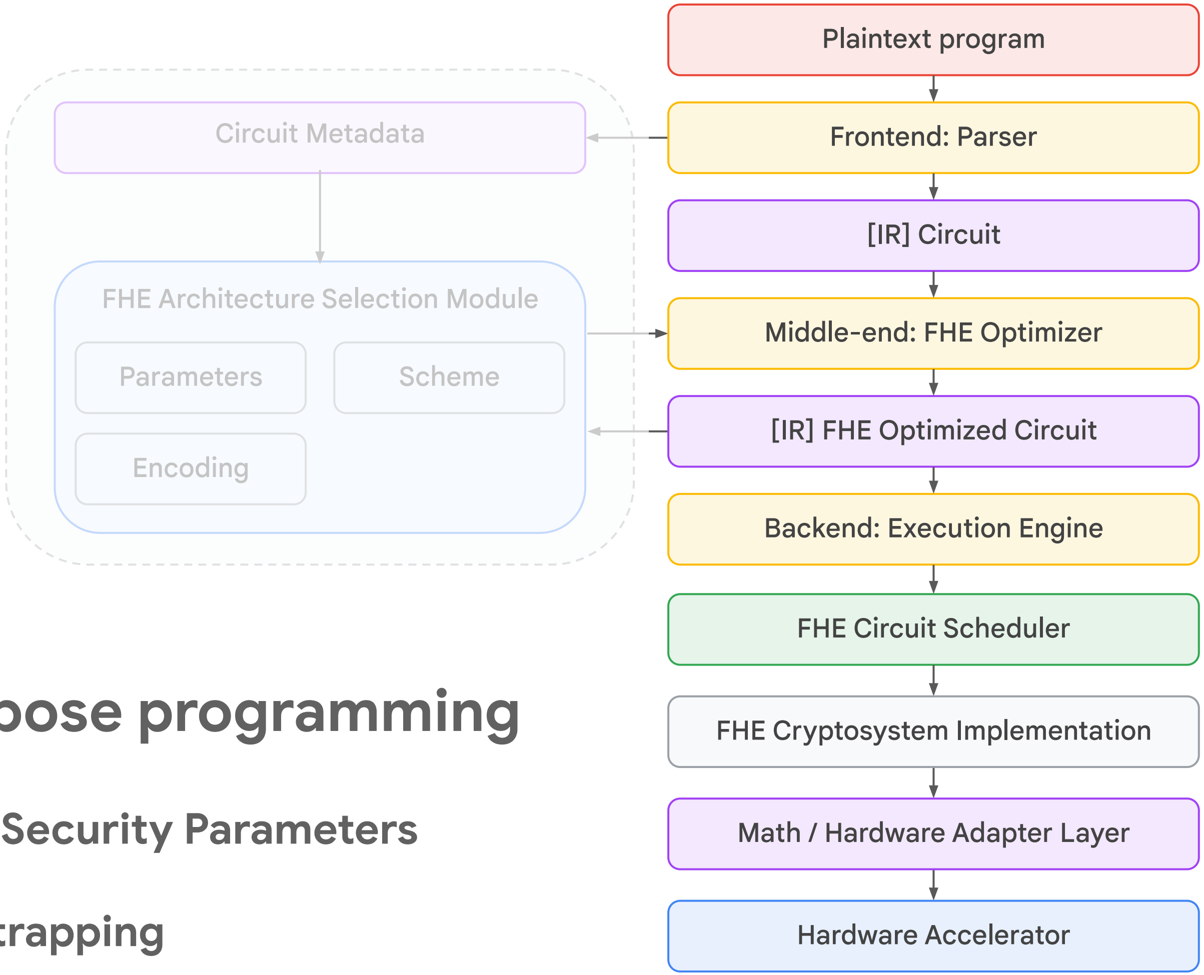
FHE Stack


Circuit Metadata
FHE Architecture Selection Module
Parameters
Scheme(s)
Encoding
Plaintext program
Frontend: Parser
[IR] Circuit
Middle-end: FHE Optimizer
[IR] FHE Optimized Circuit
Backend: Execution Engine
FHE Circuit Scheduler
FHE Cryptosystem Implementation
Math / Hardware Adapter Layer
Hardware Accelerator

Putting it all together

Modularity
Interoperability
Reusability

FHE Stack


Circuit Metadata
FHE Architecture Selection Module
Parameters
Scheme
Encoding
Plaintext program
Frontend: Parser
[IR] Circuit
Middle-end: FHE Optimizer
[IR] FHE Optimized Circuit
Backend: Execution Engine
FHE Circuit Scheduler
FHE Cryptosystem Implementation
Math / Hardware Adapter Layer
Hardware Accelerator

General purpose programming

Standard 128-bit Security Parameters
Binary encoding
TFHE gate bootstrapping

- **Plaintext program**
- Parser
- Circuit
- FHE Optimizer
- FHE Opt. Circuit
- Execution Engine
- Circuit Scheduler
- FHE Library
- Math/HW Layer
- Hardware Accelerator

Plaintext Program

String Capitalization

```
#include "string_cap.h"

#pragma hls_top
void CapitalizeString(char my_string[MAX_LENGTH]) {
 bool last_was_space = true;
#pragma hls_unroll yes
 for (int i = 0; i < MAX_LENGTH; i++) {
   char c = my_string[i];
   if (last_was_space && c >= 'a' && c <= 'z') {
     my_string[i] = c - ('a' - 'A');
   }
   last_was_space = (c == ' ');
 }
}
```

- Plaintext program
- **Parser**
- Circuit
- FHE Optimizer
- FHE Opt. Circuit
- Execution Engine
- Circuit Scheduler
- FHE Library
- Math/HW Layer
- Hardware Accelerator

Parser : Data Independent Programming

XLScc : High Level Synthesis applied to FHE

01

No if/else

(well… sort of)

02

Loops need static upper bounds

03

No branch and bound optimizations

```
if (condition) {
  return a;
} else {
  return b;
}
```

```
return
      condition.a +
      (1-condition).b;
```

Reference: SoK: Fully Homomorphic Encryption Compilers, Viand et al

Plaintext program
Parser
Circuit
FHE Optimizer
FHE Opt. Circuit
Execution Engine
Circuit Scheduler
FHE Library
Math/HW Layer
Hardware Accelerator

Circuit : XLS IR

Intermediate Representations - Modularity & Interoperability

```
package my_package
fn _ZN5State7processEh(this: (bits[1]), c: bits[8]) -> (bits[8], (bits[1])) {
 literal.20: bits[8] = literal(value=97, pos=1,10,2)
 literal.31: bits[8] = literal(value=97, pos=1,11,3)
 …
 …
 …
 ret tuple.44: (bits[8], (bits[1])) = tuple(sel.36, tuple.43, pos=1,7,1)
}

fn my_package(st: (bits[1]), c: bits[8]) -> (bits[8], (bits[1])) {
invoke.45: (bits[8], (bits[1])) = invoke(st, c, to_apply=_ZN5State7processEh,
pos=1,19,2)
 tuple_index.46: bits[8] = tuple_index(invoke.45, index=0, pos=1,19,2)
 tuple_index.47: (bits[1]) = tuple_index(invoke.45, index=1, pos=1,19,2)
 ret tuple.48: (bits[8], (bits[1])) = tuple(tuple_index.46, tuple_index.47,
pos=1,18,1)
}
```

Plaintext program

Parser

Circuit

FHE Optimizer

FHE Opt. Circuit

Execution Engine

Circuit Scheduler

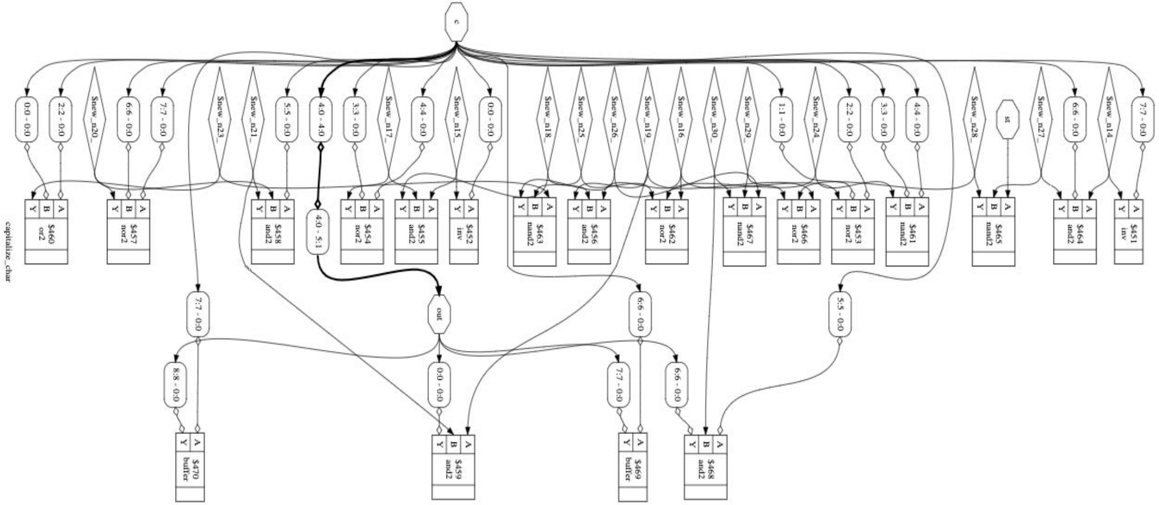
FHE Library

Math/HW Layer

Hardware Accelerator

FHE Optimizer : Netlist IR

Yosys ABC optimizations: reduces circuit size by 40 - 80%



Execution Engine : Transpiler Codegen

Server : Scheduler

```
#include <tfhe/tfhe.h>
#include <tfhe/tfhe_io.h>
#include <unordered_map>

void my_package_boolean(LweSample* result, LweSample* st, LweSample* c,
                        const TFheGateBootstrappingCloudKeySet* bk) {
  std::unordered_map<int, LweSample*> temp_nodes;

  temp_nodes[115] = new_gate_bootstrapping_ciphertext(bk->params);
  bootsCONSTANT(temp_nodes[115], 1, bk);
  ...
  bootsAND(&temp_nodes[4], temp_nodes[461], temp_nodes[256], bk);
  ...
  bootsCOPY(&result[4], temp_nodes[461], bk);
  bootsCOPY(&result[5], temp_nodes[462], bk);
  for (auto it = temp_nodes.begin(); it != temp_nodes.end(); ++it) {
    delete_gate_bootstrapping_ciphertext(it->second);
  }
}
```

- Plaintext program
- Parser
- Circuit
- FHE Optimizer
- FHE Opt. Circuit
- **Execution Engine**
- Circuit Scheduler
- FHE Library
- Math/HW Layer
- Hardware Accelerator

Execution Engine : Transpiler Codegen

Client: Encryption/Decryption API

```
#include <tfhe/tfhe.h>
#include <tfhe/tfhe_io.h>
#include <unordered_map>

absl::Status SumSimpleStruct_SCHEDULER(absl::Span<LweSample> result,
      absl::Span<const LweSample> value,
      const TFheGateBootstrappingCloudKeySet* bk);

absl::Status SumSimpleStruct(TfheRef<signed int> result
      const TfheRef<SimpleStruct> value,
      const TFheGateBootstrappingCloudKeySet* bk) {
  return SumSimpleStruct_SCHEDULER(result.get(), value.get(), bk);
}
```

```
#include <tfhe_simple_struct>

Tfhe<SimpleStruct> fhe_simple_struct(SimpleStruct(2, 3, 4));
fhe_simple_struct.SetEncrypted(simple_struct, key);
```

Plaintext program

Parser

Circuit

FHE Optimizer

FHE Opt. Circuit

Execution Engine

Circuit Scheduler

FHE Library

Math/HW Layer

Hardware Accelerator

Circuit Schedulers

Computer Architecture & Distributed Systems

- Simple Single Threaded Scheduler
- Multi-core CPU: Multi threaded Scheduler
- GPU: SIMD Scheduler
- TPU: SPMD Scheduler
- Fleet: Distributed Scheduler?

Plaintext program

Parser

Circuit

FHE Optimizer

FHE Opt. Circuit

Execution Engine

Circuit Scheduler

FHE Library

Math/HW Layer

Hardware Accelerator

FHE Library

Core Crypto Implementation

- Data Manipulation:
  - Gates, ADD, MUL, Look Up Table Evaluation
- Ciphertext Maintenance
  - Bootstrap, Rescale, Modulus Switch
  - Key Switch, Scheme Switch
- Specification for Support
  - Message packing
  - Security parameters

Plaintext program

Parser

Circuit

FHE Optimizer

FHE Opt. Circuit

Execution Engine

Circuit Scheduler

FHE Library

Math/HW Layer

Hardware Accelerator

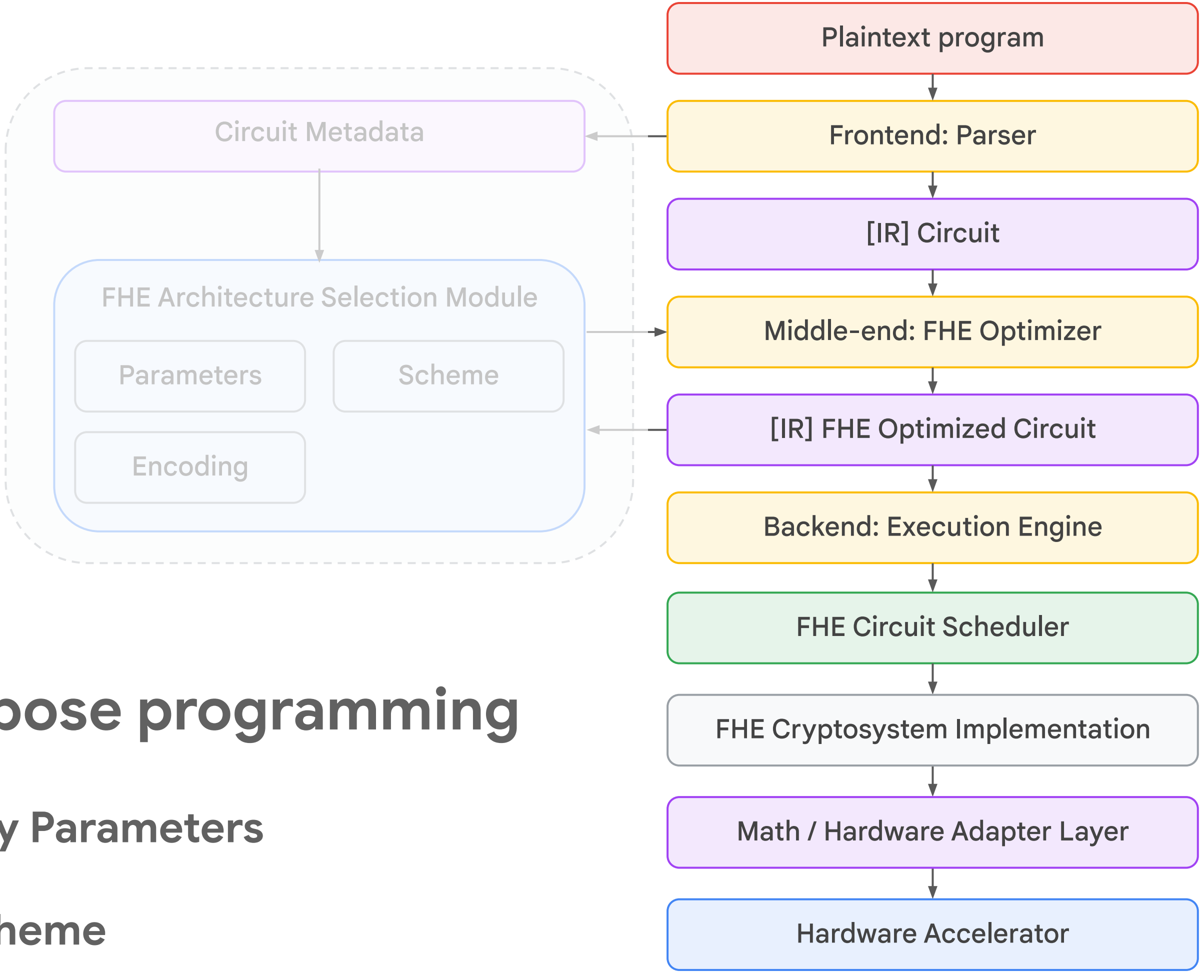
Micro Instruction Scheduler

Computer Architecture & Hardware Co-design

- Hardware optimizations for primitives
- Accelerate Math Primitive: Polynomial Multiplication
- Accelerate Crypto Primitives: Bootstrap, etc

- Programs are not 100% parallalizable
- How to translate amortized primitives speedup → application speedup ?

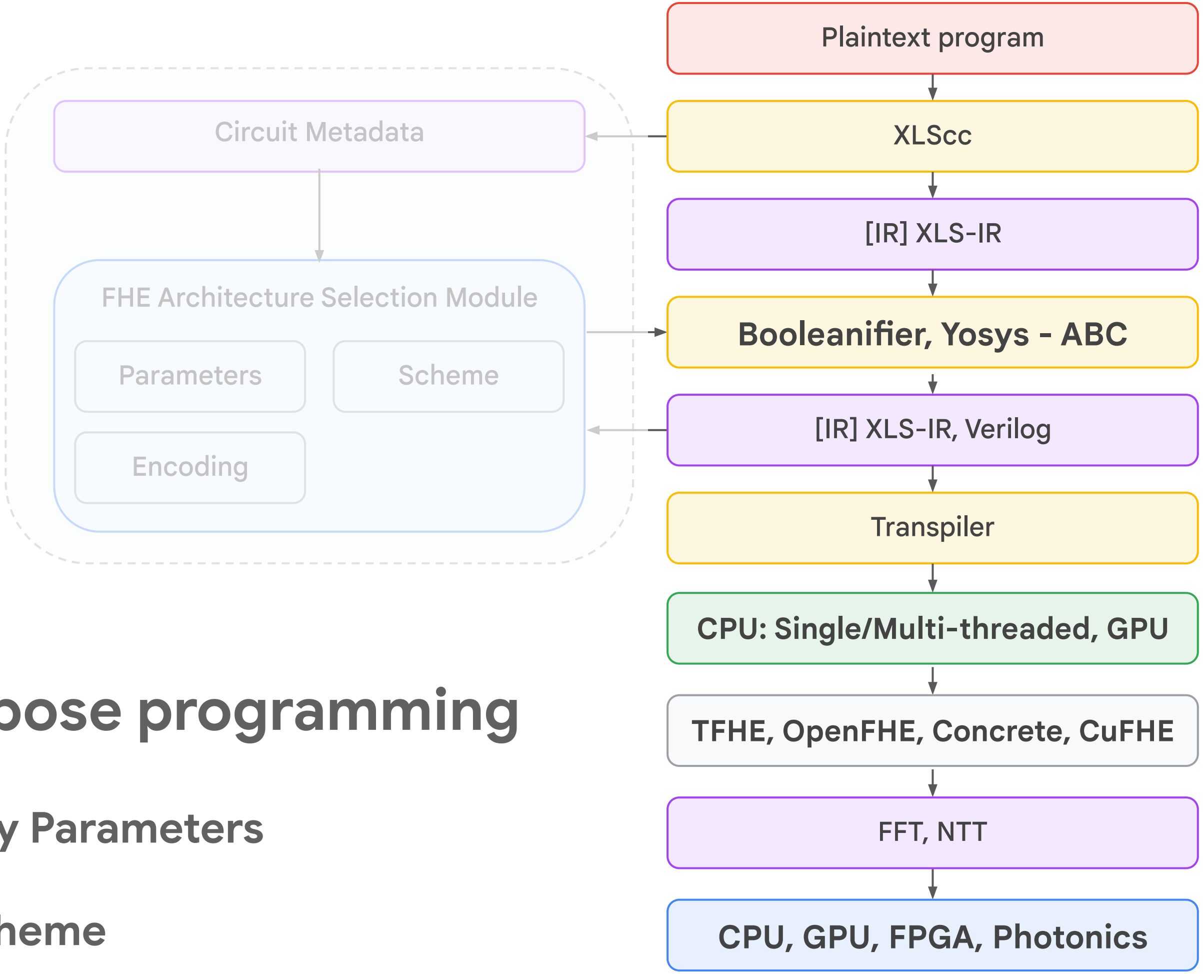
FHE Stack


Circuit Metadata
FHE Architecture Selection Module
Parameters
Scheme
Encoding
Plaintext program
Frontend: Parser
[IR] Circuit
Middle-end: FHE Optimizer
[IR] FHE Optimized Circuit
Backend: Execution Engine
FHE Circuit Scheduler
FHE Cryptosystem Implementation
Math / Hardware Adapter Layer
Hardware Accelerator

General purpose programming

Standard Security Parameters
Binary encoding
TFHE Boolean Scheme

FHE Stack


Circuit Metadata
FHE Architecture Selection Module
Parameters
Scheme
Encoding
Plaintext program
XLScc
[IR] XLS-IR
Booleanifier, Yosys - ABC
[IR] XLS-IR, Verilog
Transpiler
CPU: Single/Multi-threaded, GPU
TFHE, OpenFHE, Concrete, CuFHE
FFT, NTT
CPU, GPU, FPGA, Photonics

General purpose programming

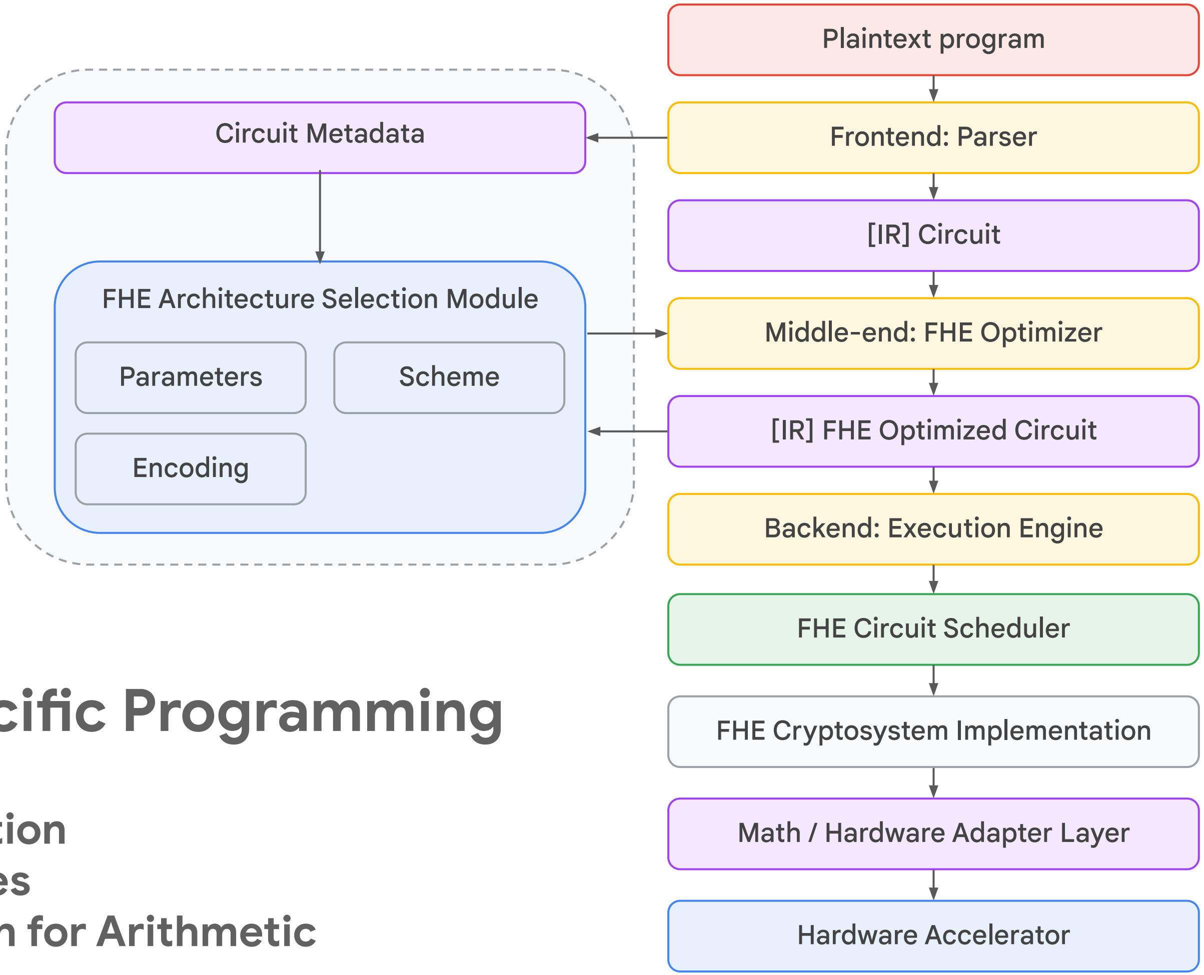
Standard Security Parameters
Binary encoding
TFHE Boolean Scheme

FHE Stack : End to End Metrics

Staring Capitalization (32 characters)

Hardware	Scheme/Library	Bootstrap (ms)	Optimizer	Circuit size	Scheduler	Time (s)
CPU	CGGI/TFHE	13	XLS	920	Single	53.23
CPU - 96 core	CGGI/TFHE	13	XLS	920	Multi-threaded	1.84
CPU - 96 core	**CGGI/TFHE**	**13**	**Yosys**	**537**	**Multi-threaded**	**0.9**
CPU	DM/BinFHE	58	XLS	920	Single	274
CPU - 96 core	DM/BinFHE	58	XLS	920	Multi-threaded	7.4
CPU - 96 core	**DM/BinFHE**	**58**	**Yosys**	**537**	**Multi-threaded**	**4.4**
CPU	CGGI/TFHE-rs	12.5	Yosys	537	?	?
GPU	CGGI/CuFHE	0.5*	Yosys	537	?	?
FPGA	CGGI/KU Leuven	0.03*	Yosys / ?	537	?	?

FHE Stack

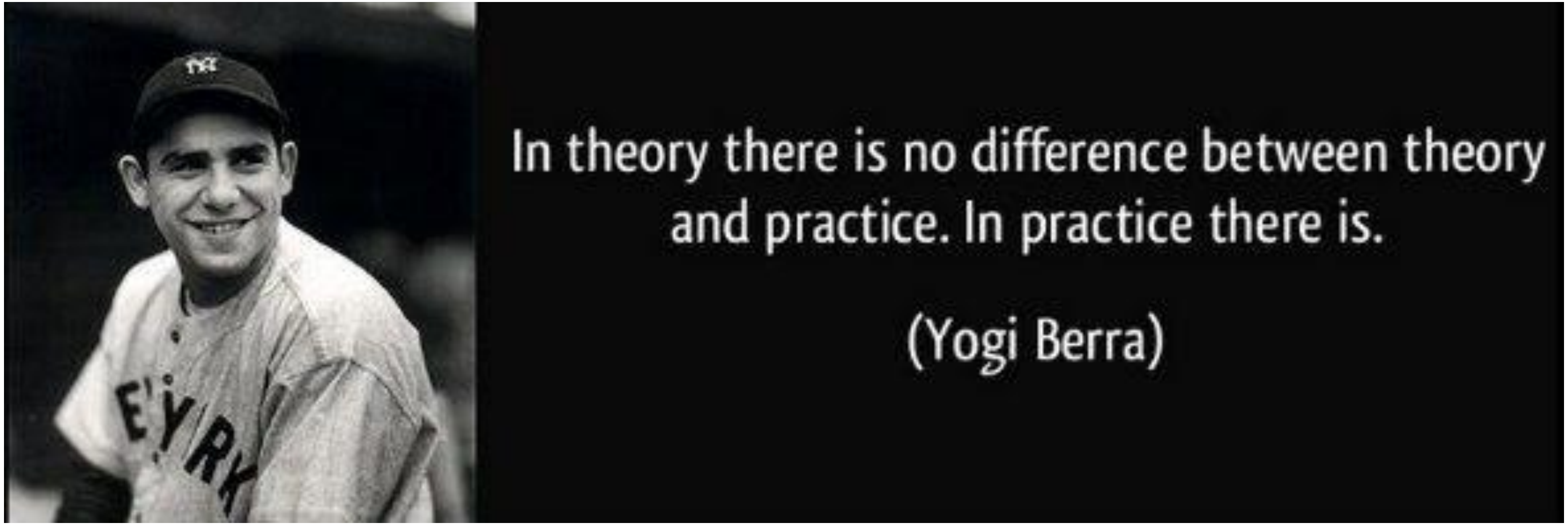

Circuit Metadata
FHE Architecture Selection Module
Parameters
Scheme
Encoding
Plaintext program
Frontend: Parser
[IR] Circuit
Middle-end: FHE Optimizer
[IR] FHE Optimized Circuit
Backend: Execution Engine
FHE Circuit Scheduler
FHE Cryptosystem Implementation
Math / Hardware Adapter Layer
Hardware Accelerator

Domain Specific Programming

Parameter Selection
Packing strategies
Scheme Selection for Arithmetic

ARE WE
HAVING
FHE YET?

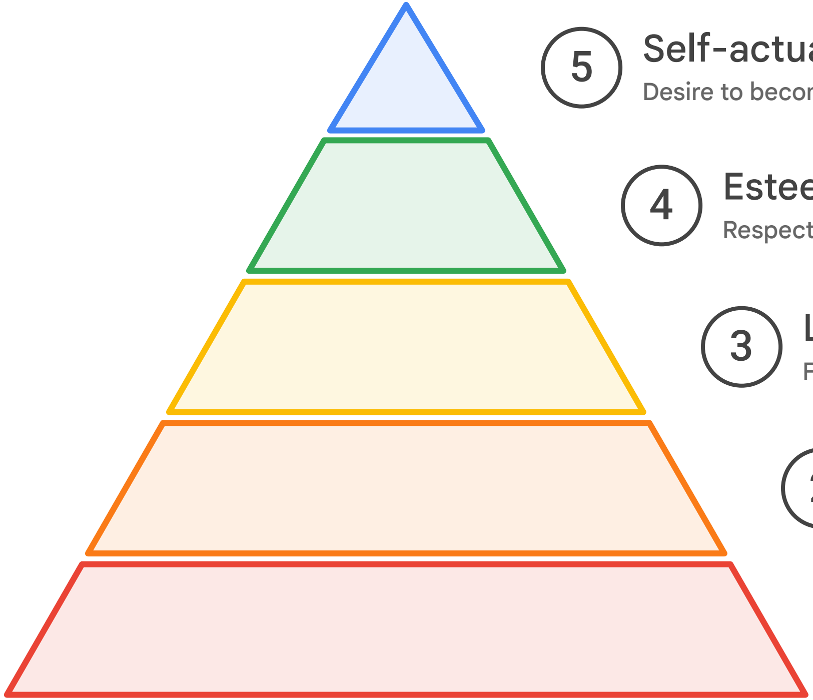
Applied FHE for privacy



In theory there is no difference between theory and practice. In practice there is.

(Yogi Berra)

Maslow’s Hierarchy of Needs



5 **Self-actualization**
Desire to become the most that one can be

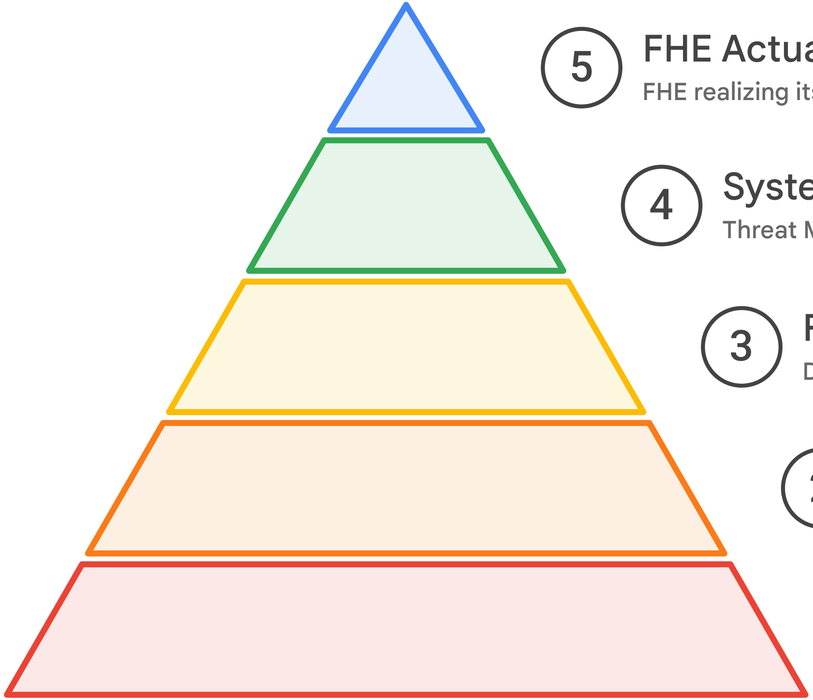
4 **Esteem**
Respect, self-esteem, status, recognition, strength, freedom

3 **Love and belonging**
Friendship, intimacy, family, sense of connection

2 **Safety needs**
Personal security, employment, resources, health, property

1 **Physiological needs**
Air, water, food, shelter, sleep, clothing, reproduction

FHE Hierarchy of Needs



5 **FHE Actualization**
FHE realizing its true potential as a PET

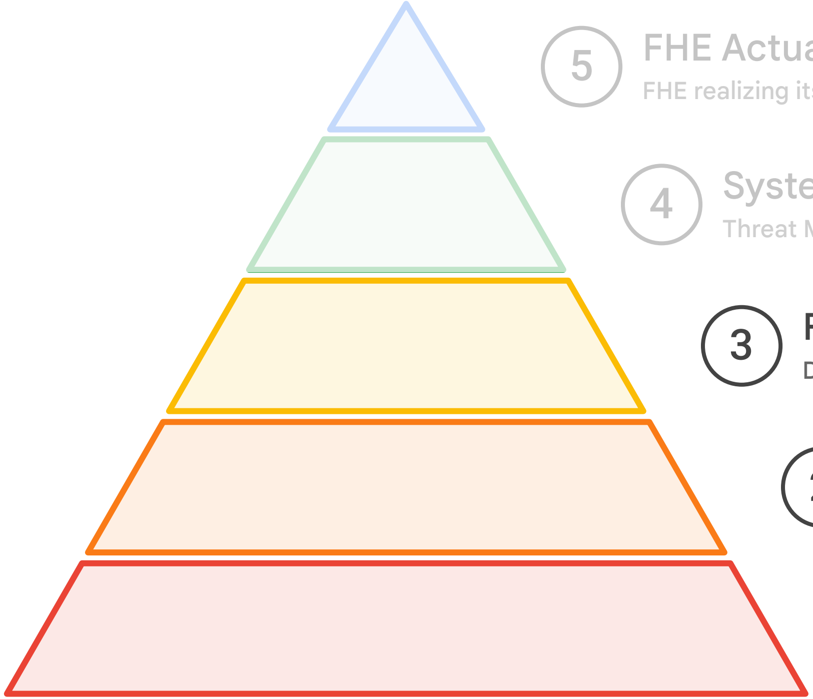
4 **Systems integration / Privacy engineering**
Threat Modeling, Privacy leaks, Network bandwidth, Observability

3 **FHE application development**
Development speed, Scheme Interoperability, Debugging

2 **Compilers / Transpilers**
Param Selection, Optimizers, Schedulers, Intermediate Representations

1 **FHE Instruction Set Architecture**
Data Manipulation, Ciphertext Maintenance, Hardware Cost

FHE Hierarchy of Needs : Deficiency Needs



5 **FHE Actualization**

FHE realizing its true potential as a PET

4 **Systems integration / Privacy engineering**

Threat Modeling, Privacy leaks, Network bandwidth, Observability

3 **FHE application development**

Development speed, Scheme Interoperability, Debugging

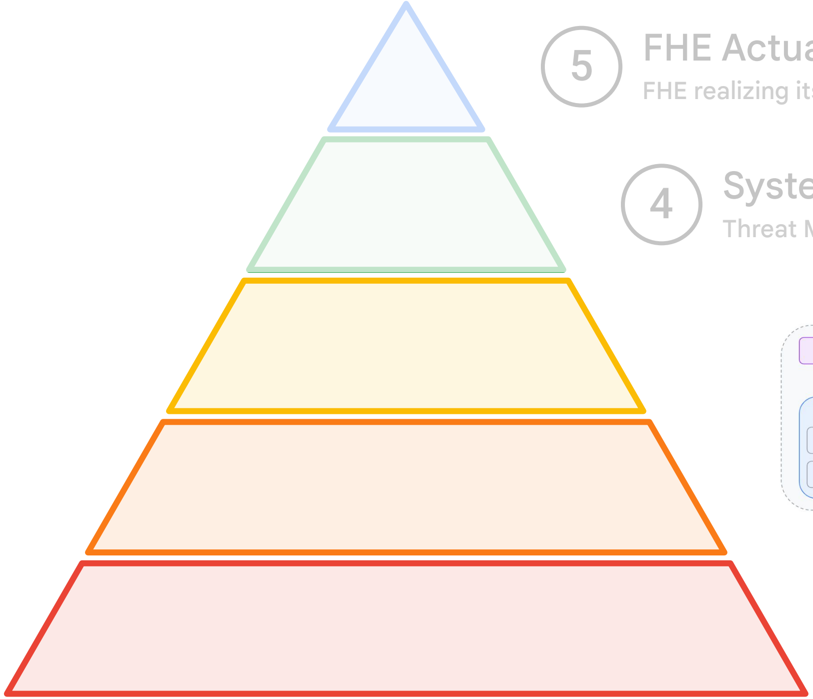
2 **Compilers / Transpilers**

Param Selection, Optimizers, Schedulers, Intermediate Representations

1 **FHE Instruction Set Architecture**

Data Manipulation, Ciphertext Maintenance, Hardware Cost

FHE Hierarchy of Needs: Deficiency Needs

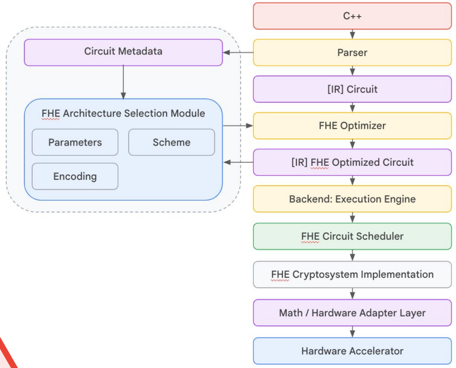


5 **FHE Actualization**

FHE realizing its true potential as a PET

4 **Systems integration / Privacy engineering**

Threat Modeling, Privacy leaks, Network bandwidth, Observability



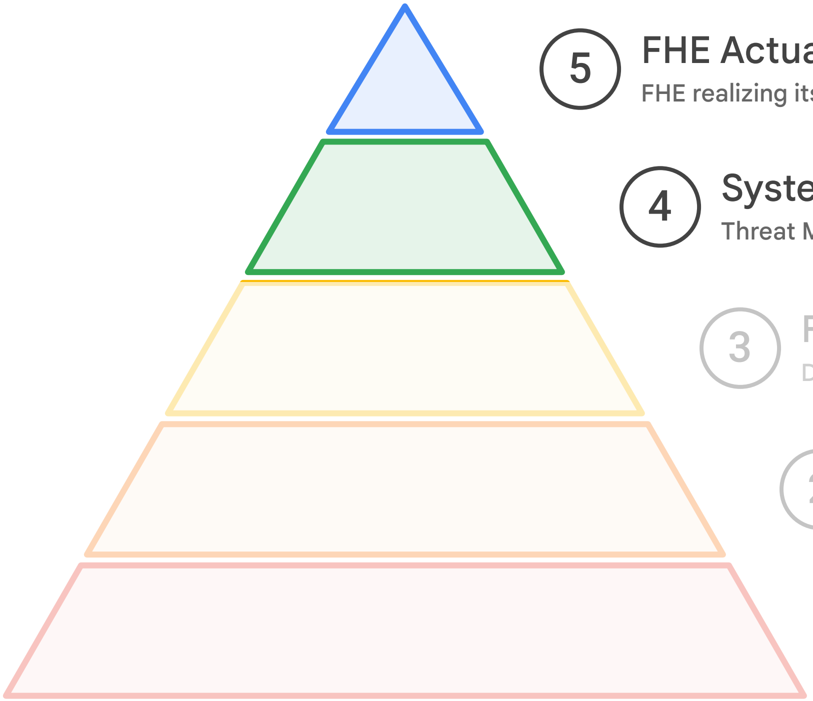
C++
Parser
[IR] Circuit
FHE Optimizer
[IR] FHE Optimized Circuit
Backend: Execution Engine
FHE Circuit Scheduler
FHE Cryptosystem Implementation
Math / Hardware Adapter Layer
Hardware Accelerator
Circuit Metadata
FHE Architecture Selection Module
Parameters
Scheme
Encoding

FHE Stack

The ultimate success of Cryptography lies in ***kicking the cryptographer out of the loop.***

- Moti Yung

FHE Hierarchy of Needs: Growth Needs



5 **FHE Actualization**
FHE realizing its true potential as a PET

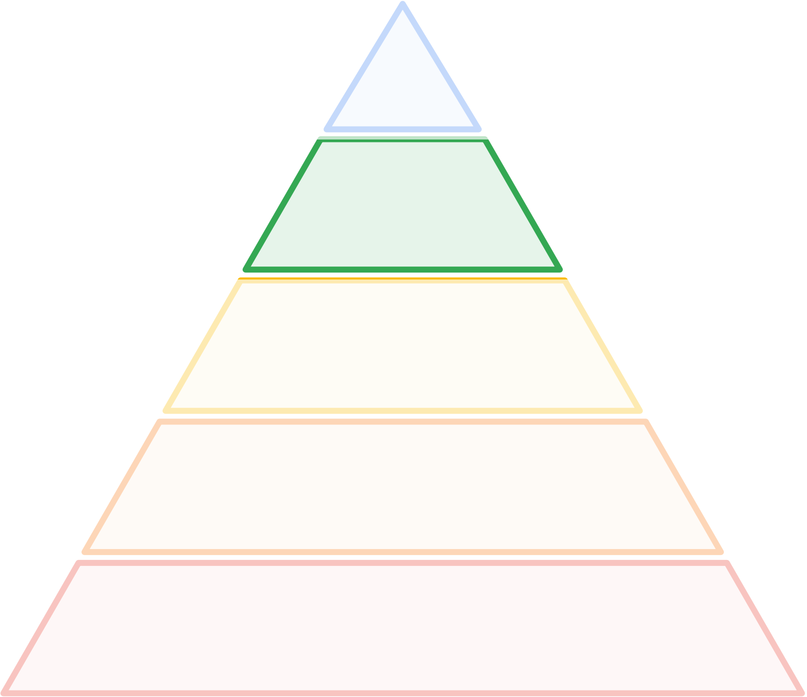
4 **Systems integration / Privacy engineering**
Threat Modeling, Privacy leaks, Network bandwidth, Observability

3 **FHE application development**
Development speed, Scheme Interoperability, Debugging

2 **Compilers / Transpilers**
Param Selection, Optimizers, Schedulers, Intermediate Representations

1 **FHE Instruction Set Architecture**
Data Manipulation, Ciphertext Maintenance, Hardware Cost

4. Systems Integration / Privacy Engineering



Key Management

Private key, public key, multi key, key backups

Trust Model and Protocol Development

FHE combined with Privacy Enhancing Technologies

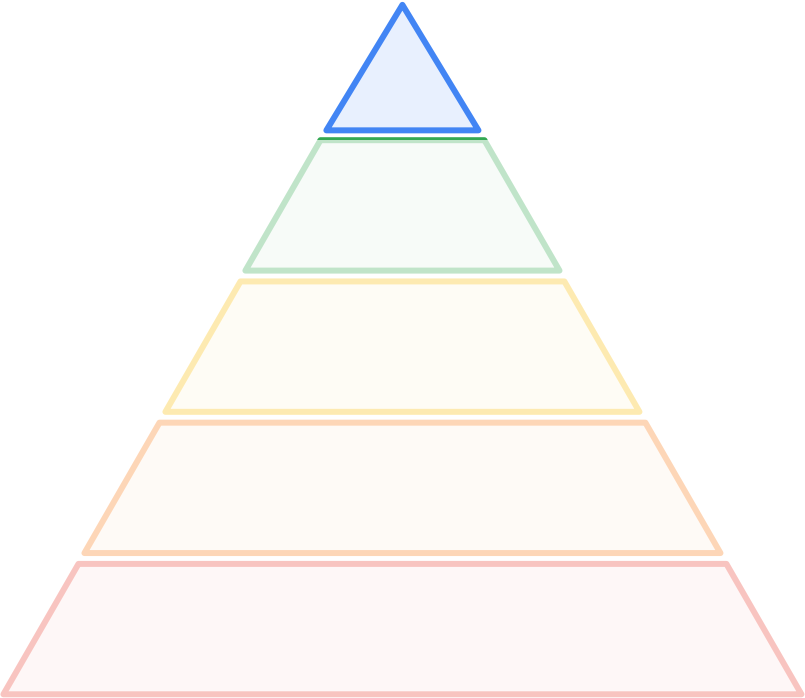
Privacy Leaks in Data Size

Probabilistic sizing to optimize for performance

Open Problem: Ciphertext Expansion

Packing and hybrid homomorphic encryption

5. FHE Actualization



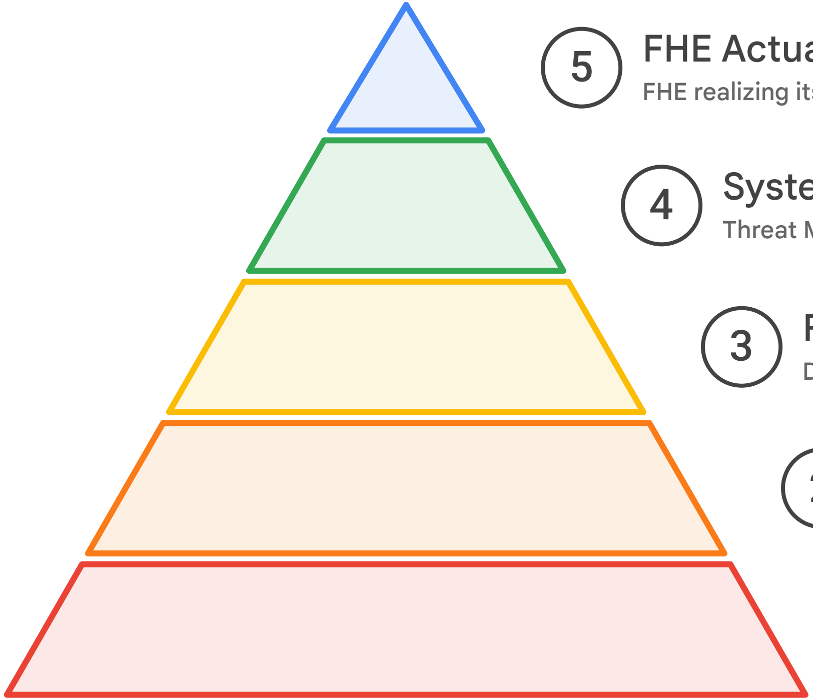
Latency / Network Bandwidth / Privacy
Metrics

$$$ for Hardware
Budget

Product Timelines
Launch dates

Software Developer Hours
Engineering time

FHE Hierarchy of Needs



5 **FHE Actualization**
FHE realizing its true potential as a PET

4 **Systems integration / Privacy engineering**
Threat Modeling, Privacy leaks, Network bandwidth, Observability

3 **FHE application development**
Development speed, Scheme Interoperability, Debugging

2 **Compilers / Transpilers**
Param Selection, Optimizers, Schedulers, Intermediate Representations

1 **FHE Instruction Set Architecture**
Data Manipulation, Ciphertext Maintenance, Hardware Cost

What’s Next

1. Unify Compilers with IR (MLIR)
2. Comprehensive Benchmarking Set
3. Find that one demonstrable FHE use case

Thank you!

google/fully-homomorphic-encryption

[Discussion Forum]

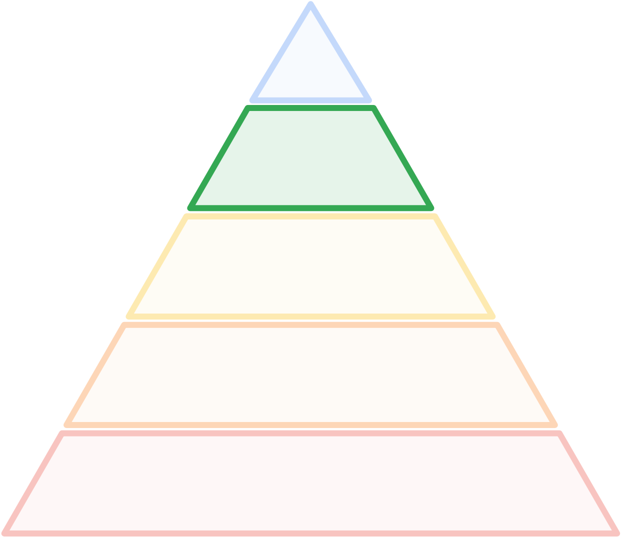
fhe-open-source-users@google.com

fhe-open-source@google.com
gshruthi@google.com

The END

4.2 Privacy Engineering

Privacy Leaks in Data Size



Leak Size in Plaintext

Minimum privacy

Bucket Size

S, M, L

Open Problem: Probabilistic Sizing

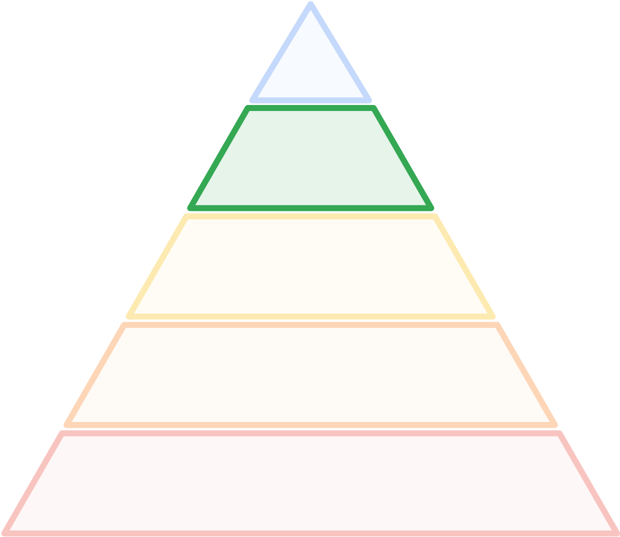
Quantifying privacy

Max Size

Maximum privacy

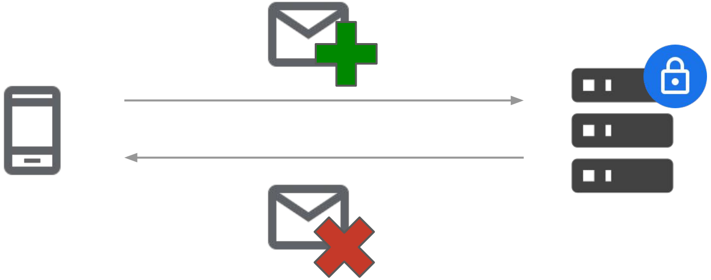
4.3 Systems Integration

Ciphertext Expansion



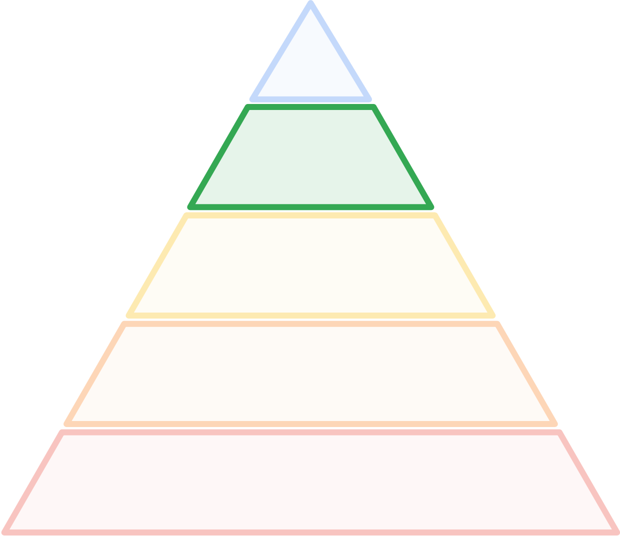
Does Hybrid Homomorphic Encryption solve everything?

What about ciphertext expansion of the result from server to client



4.1 Privacy Engineering

Trust Model



Adversarial Server

Zero Knowledge Proofs

Adversarial Client

Differential privacy

Multi-Party Aggregation with Honest-Curious

Secure Multi Party Computation