ZAMA

5th HomomorphicEncryption.org Standards Meeting
Geneva, Switzerland

An overview of the Concrete Framework

Quentin Bourgerie	quentin.bourgerie@zama.ai
Ilaria Chillotti	**ilaria.chillotti@zama.ai**
Damien Ligier	**damien.ligier@zama.ai**
Jean-Baptiste Orfila	jb.orfila@zama.ai
Alexandre Quint	alexandre.quint@zama.ai
Samuel Tap	samuel.tap@zama.ai

SEPTEMBER 1, 2022

The goal

Why are we here?

The goal

Why are we here?

To boost large-scale FHE adoption

The goal

Why are we here?

To boost large-scale FHE adoption

Organizations

- homomorphicencryption.org
- fhe.org

The goal

Why are we here?

To boost large-scale FHE adoption

Organizations
- homomorphicencryption.org
- fhe.org

Workshops
- WAHC
- fhe.org

The goal

Why are we here?

To boost large-scale FHE adoption

Organizations
- homomorphicencryption.org
- fhe.org

Workshops
- WAHC
- fhe.org

Competitions
- iDASH since 2014
- fhe.org competition 2022

The goal

Why are we here?

To boost large-scale FHE adoption

Organizations
- homomorphicencryption.org
- fhe.org

Workshops
- WAHC
- fhe.org

ISO standardization

Competitions
- iDASH since 2014
- fhe.org competition 2022

The goal

Why are we here?

To boost large-scale FHE adoption

Organizations
- homomorphicencryption.org
- fhe.org

Workshops
- WAHC
- fhe.org

ISO standardization

Competitions
- iDASH since 2014
- fhe.org competition 2022

Research community

The goal

Why are we here?

To boost large-scale FHE adoption

Organizations
- homomorphicencryption.org
- fhe.org

Workshops
- WAHC
- fhe.org

ISO standardization

Competitions
- iDASH since 2014
- fhe.org competition 2022

Research community

Lattice estimator (LWE estimator)

The goal

Why are we here?

To boost large-scale FHE adoption

Organizations
- homomorphicencryption.org
- fhe.org

Workshops
- WAHC
- fhe.org

ISO standardization

Competitions
- iDASH since 2014
- fhe.org competition 2022

Research community

Lattice estimator (LWE estimator)

Many open source FHE solutions

The goal

Why are we here?

To boost large-scale FHE adoption

Organizations
- homomorphicencryption.org
- fhe.org

Workshops
- WAHC
- fhe.org

ISO standardization

Competitions
- iDASH since 2014
- fhe.org competition 2022

Research community

Lattice estimator (LWE estimator)

Many open source FHE solutions

JoC 2022: Topical Collection Computing on Encrypted Data

The goal

Why are we here?

To boost large-scale FHE adoption

Organizations
- homomorphicencryption.org
- fhe.org

Workshops
- WAHC
- fhe.org

Competitions
- iDASH since 2014
- fhe.org competition 2022

Many open source FHE solutions

ISO standardization

Research community

Lattice estimator (LWE estimator)

JoC 2022: Topical Collection Computing on Encrypted Data

Community effort

Join forces

Obstacles

More to overcome?

Obstacles

More to overcome?

Which features do we need?

Efficiency

Security

Usability

Obstacles

More to overcome?

Which features do we need?

Efficiency

Security

Usability

Users will be non-FHE expert developers
- Crypto libraries will not be sufficient

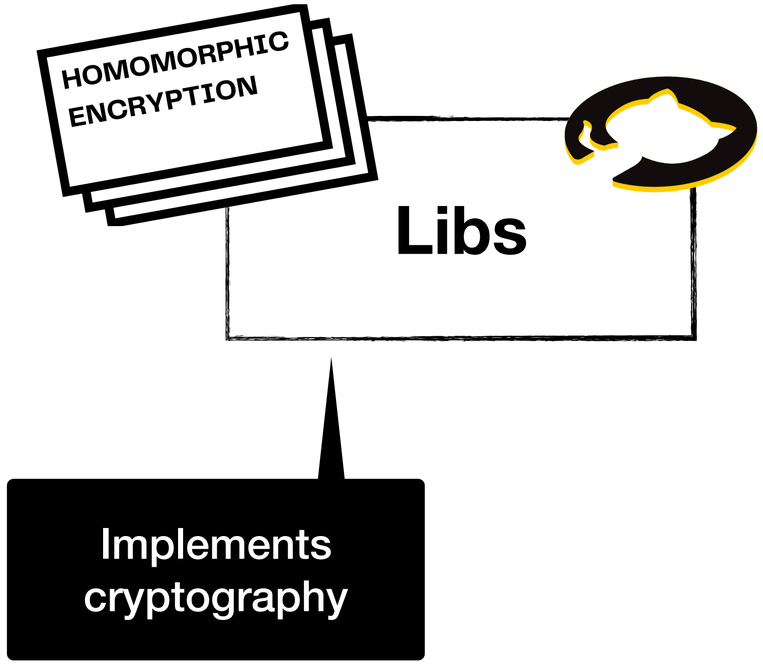
How to choose crypto parameters
- Still a big open problem

CFW

Concrete Framework

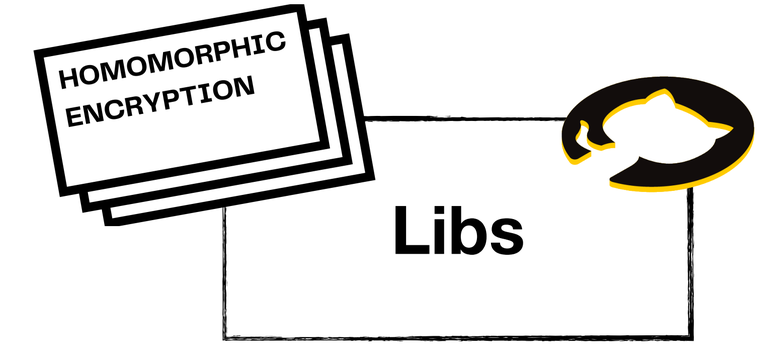
CFW

Concrete Framework



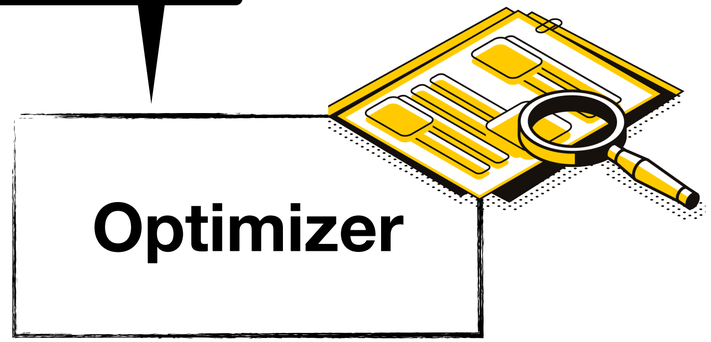
CFW

Concrete Framework



Implements cryptography

Finds best parameters and topology



CFW

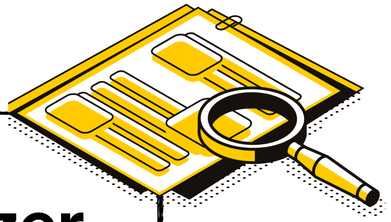
Concrete Framework

HOMOMORPHIC ENCRYPTION

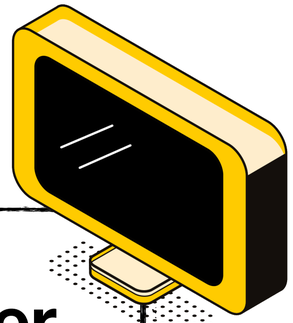
Libs

Implements cryptography

Finds best parameters and topology



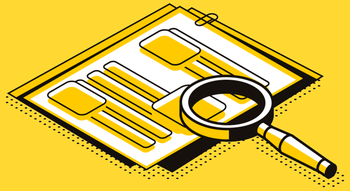
Optimizer



Compiler

Lowers from popular languages

Libs

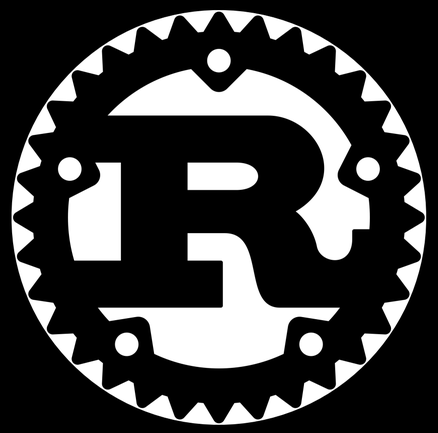


Meetup: Homomorphic Large Precision Integers Using Concrete

https://fhe.org/meetups/homomorphic-Large-Precision-Integers-Using-Concrete

Concrete

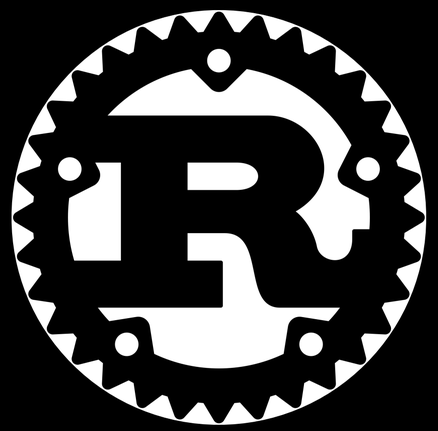
Several libraries

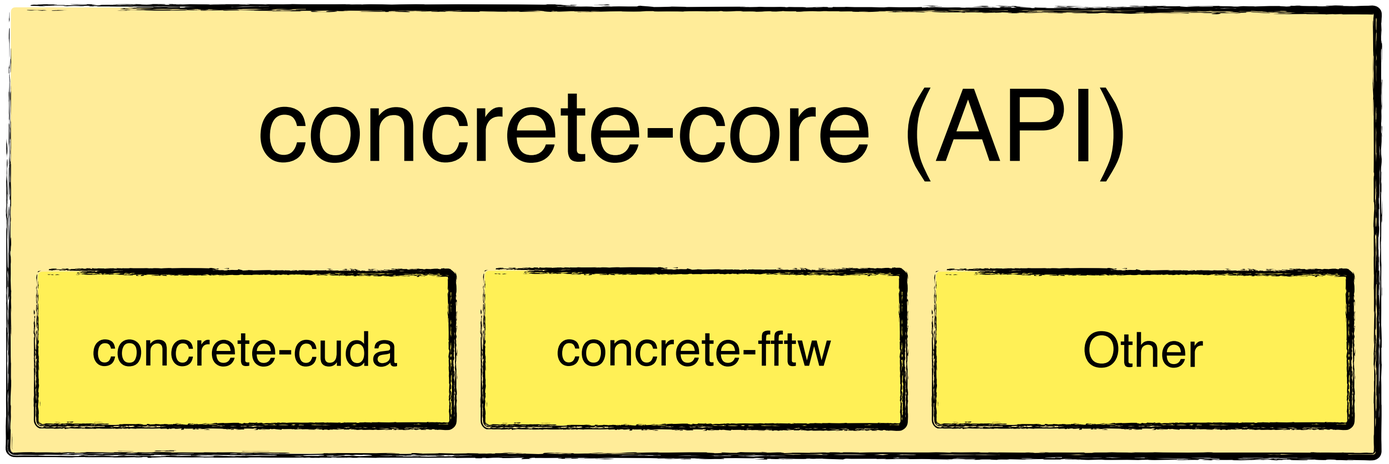


concrete-core (API)

Concrete

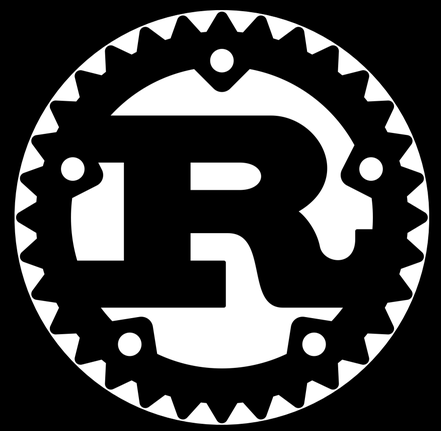
Several libraries

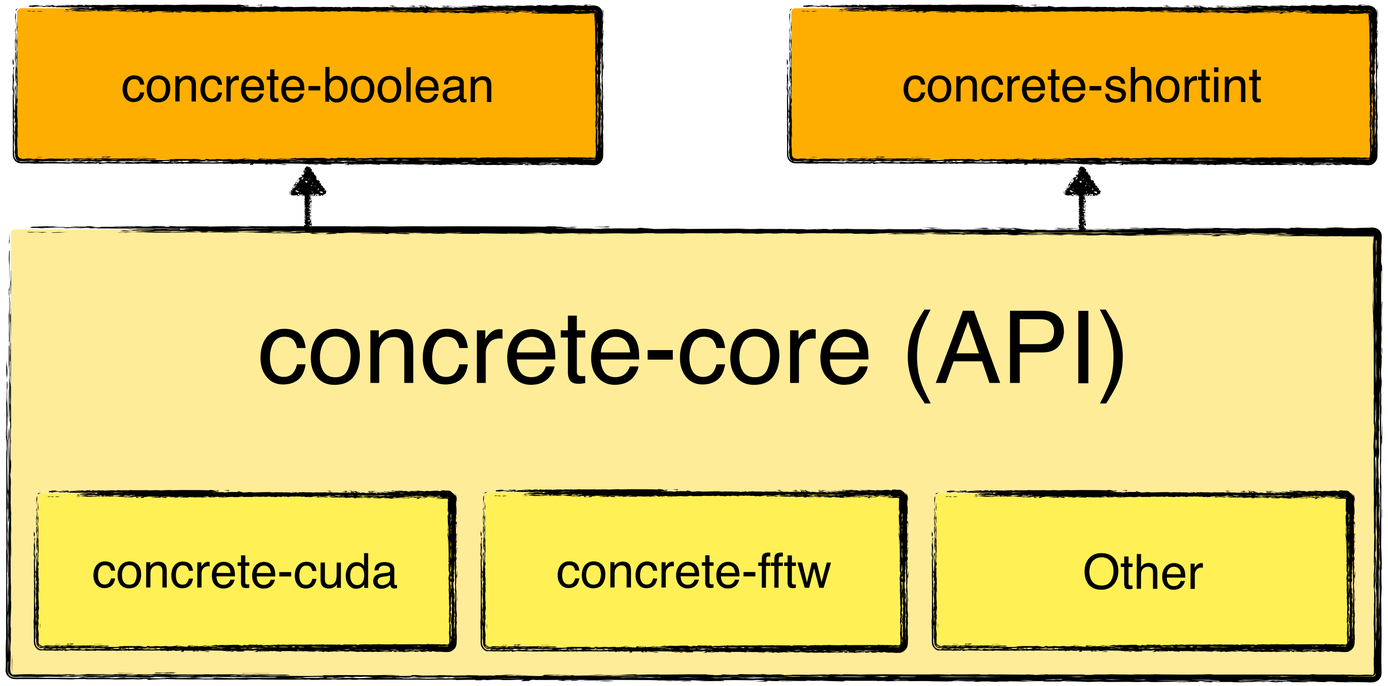




Concrete

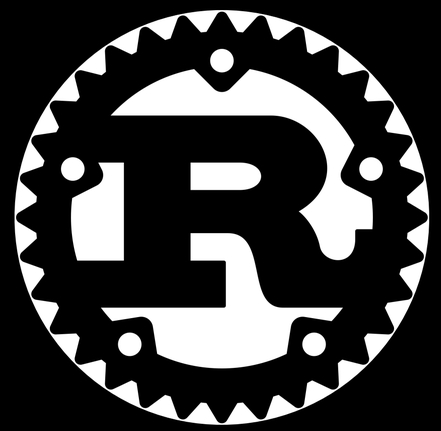
Several libraries

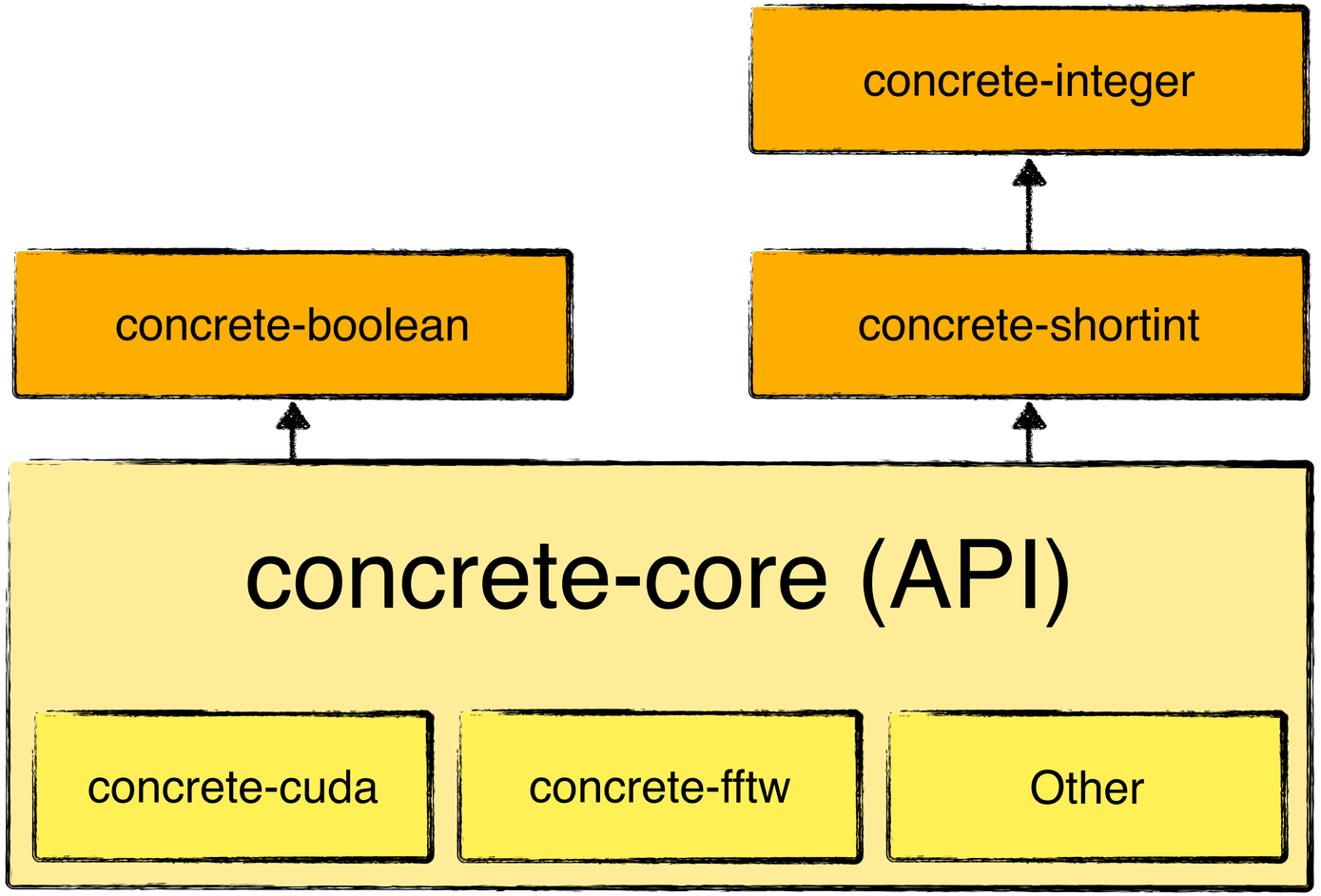




Concrete

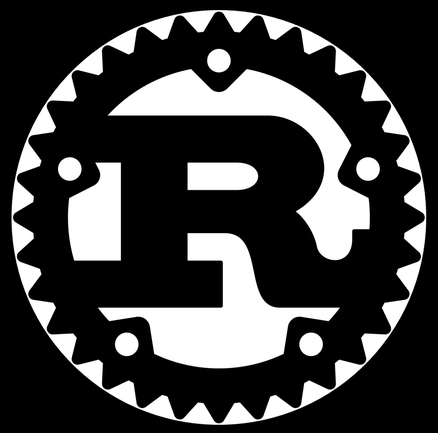
Several libraries

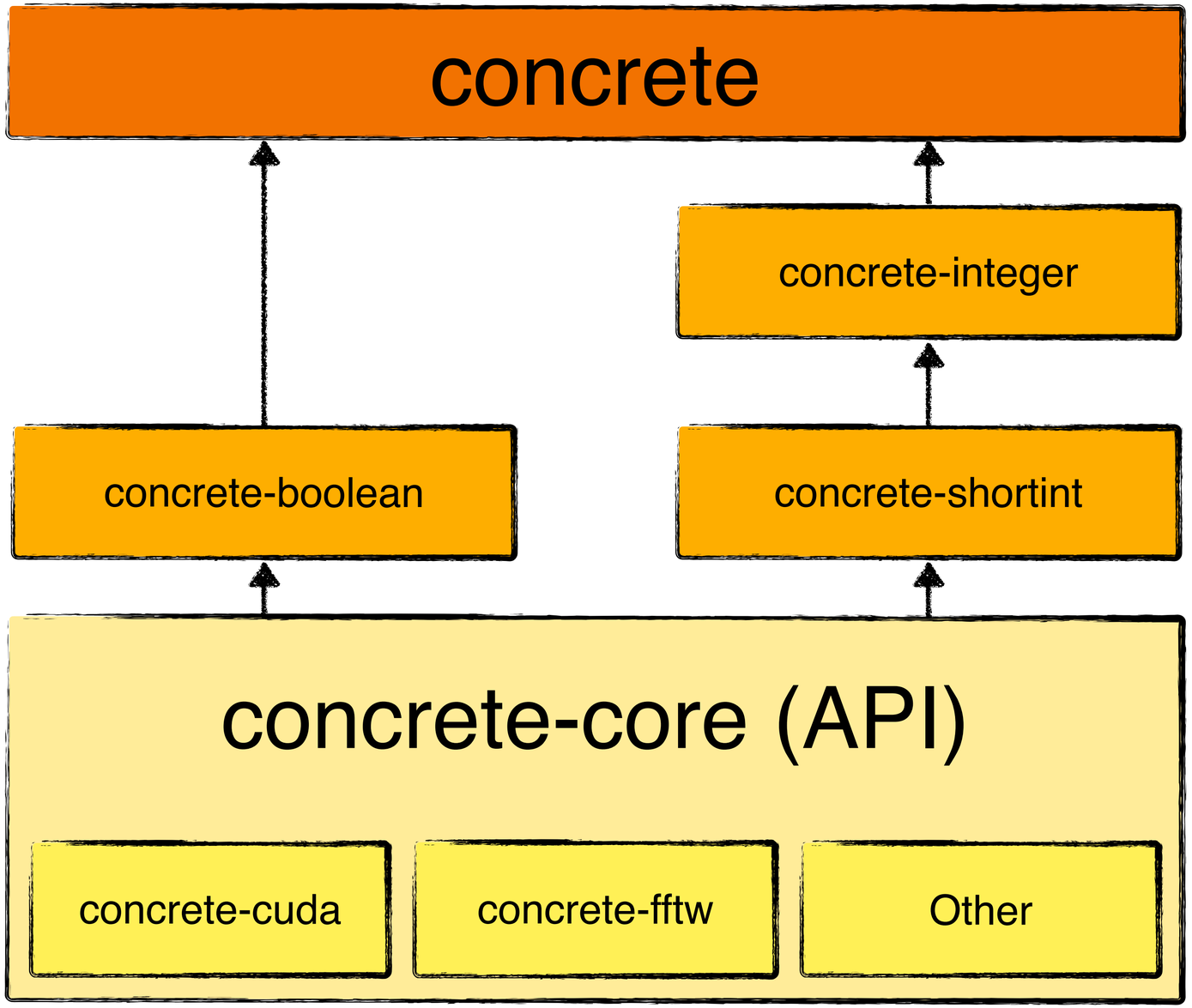




Concrete

Several libraries





3 Libs

3 data types

3 Libs

3 data types

Boolean

- 1 binary message in 1 ciphertext
- NOT, OR, AND, XOR, …

3 Libs

3 data types

Boolean

- 1 binary message in 1 ciphertext
- NOT, OR, AND, XOR, …

ShortInt

- 1 modular integer (up to 8-bits of precision) in 1 ciphertext
- Add, Sub, Mul, Mod, LUT Evaluation, …

3 Libs

3 data types

Boolean

- 1 binary message in 1 ciphertext
- NOT, OR, AND, XOR, …

ShortInt

- 1 modular integer (up to 8-bits of precision) in 1 ciphertext
- Add, Sub, Mul, Mod, LUT Evaluation, …

Integer

- 1 large integer (CRT or Radix representation) in several ciphertexts
- Add, Sub, Mul, Mod, LUT Evaluation, …

3 Libs

3 data types

Boolean

- 1 binary message in 1 ciphertext
- NOT, OR, AND, XOR, …

Sho

- 1
- Add

Several parameter sets:

Trade off between
execution cost & error probability

Integer

- 1 large integer (CRT or Radix representation) in several ciphertexts
- Add, Sub, Mul, Mod, LUT Evaluation, …

Concrete

A complete Rust library

```rust
use concrete::{ConfigBuilder, generate_keys, set_server_key, FheUint8};
use concrete::prelude::*;

fn main() {
    let config = ConfigBuilder::all_disabled()
        .enable_default_uint8()
        .build();

    let (client_key, server_key) = generate_keys(config);

    set_server_key(server_key);

    let clear_a = 27u8;
    let clear_b = 128u8;

    let a = FheUint8::encrypt(clear_a, &client_key);
    let b = FheUint8::encrypt(clear_b, &client_key);

    let result = a + b;

    let decrypted_result: u8 = result.decrypt(&client_key);

    let clear_result = clear_a + clear_b;

    assert_eq!(decrypted_result, clear_result);
}
```

Concrete

A complete Rust library

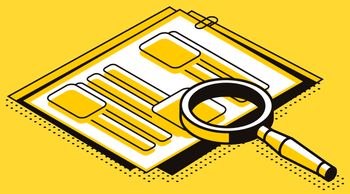
```rust
use concrete::{ConfigBuilder, generate_keys, set_server_key, FheUint8};
use concrete::prelude::*;

fn main() {
    let config = ConfigBuilder::all_disabled()
        .enable_default_uint8()
        .build();

    let (client_key, server_key) = generate_keys(config);

    set_server_key(server_key);

    let clear_a = 27u8;
    let clear_b = 128u8;

    let a = FheUint8::encrypt(clear_a, &client_key);
    let b = FheUint8::encrypt(clear_b, &client_key);

    let result = a + b;

    let decrypted_result: u8 = result.decrypt(&client_key);

    let clear_result = clear_a + clear_b;

    assert_eq!(decrypted_result, clear_result);
}
```

Concrete

A complete Rust library

```rust
use concrete::{ConfigBuilder, generate_keys, set_server_key, FheUint8};
use concrete::prelude::*;

fn main() {
    let config = ConfigBuilder::all_disabled()
        .enable_default_uint8()
        .build();

    let (client_key, server_key) = generate_keys(config);

    set_server_key(server_key);

    let clear_a = 27u8;
    let clear_b = 128u8;

    let a = FheUint8::encrypt(clear_a, &client_key);
    let b = FheUint8::encrypt(clear_b, &client_key);

    let result = a + b;

    let decrypted_result: u8 = result.decrypt(&client_key);

    let clear_result = clear_a + clear_b;

    assert_eq!(decrypted_result, clear_result);
}
```

Optimizer



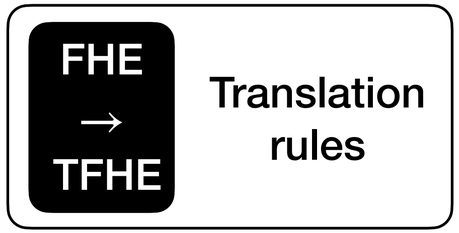
Parameter Optimization and Larger Precision for (T)FHE
https://eprint.iacr.org/2022/704.pdf

Optimizer

Overview

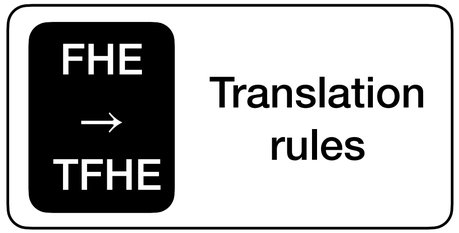
Optimizer

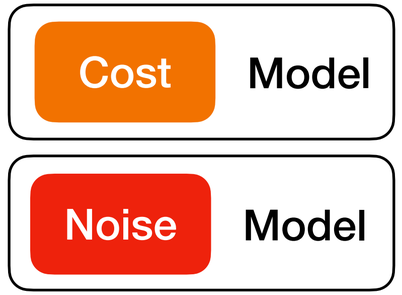
Overview



Optimizer

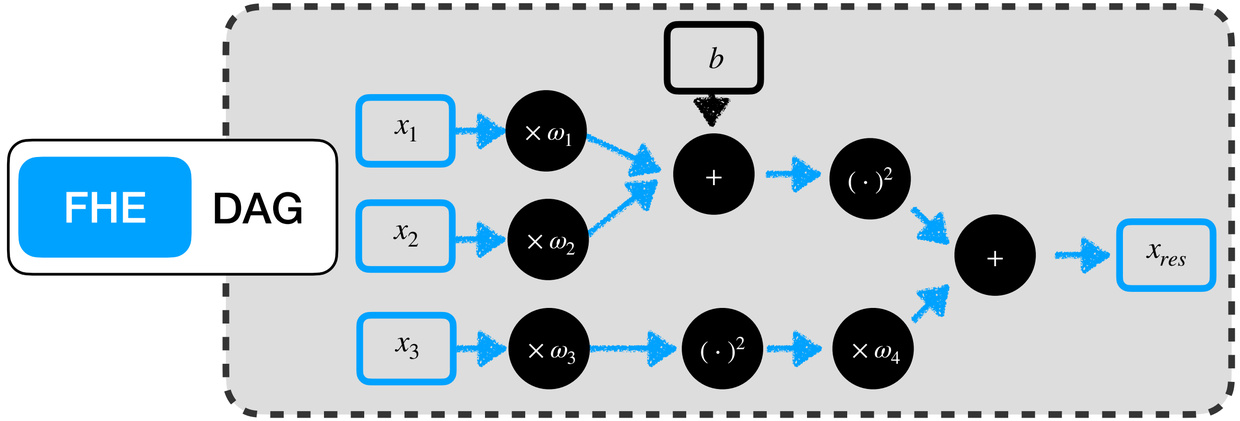
Overview





Optimizer

Overview



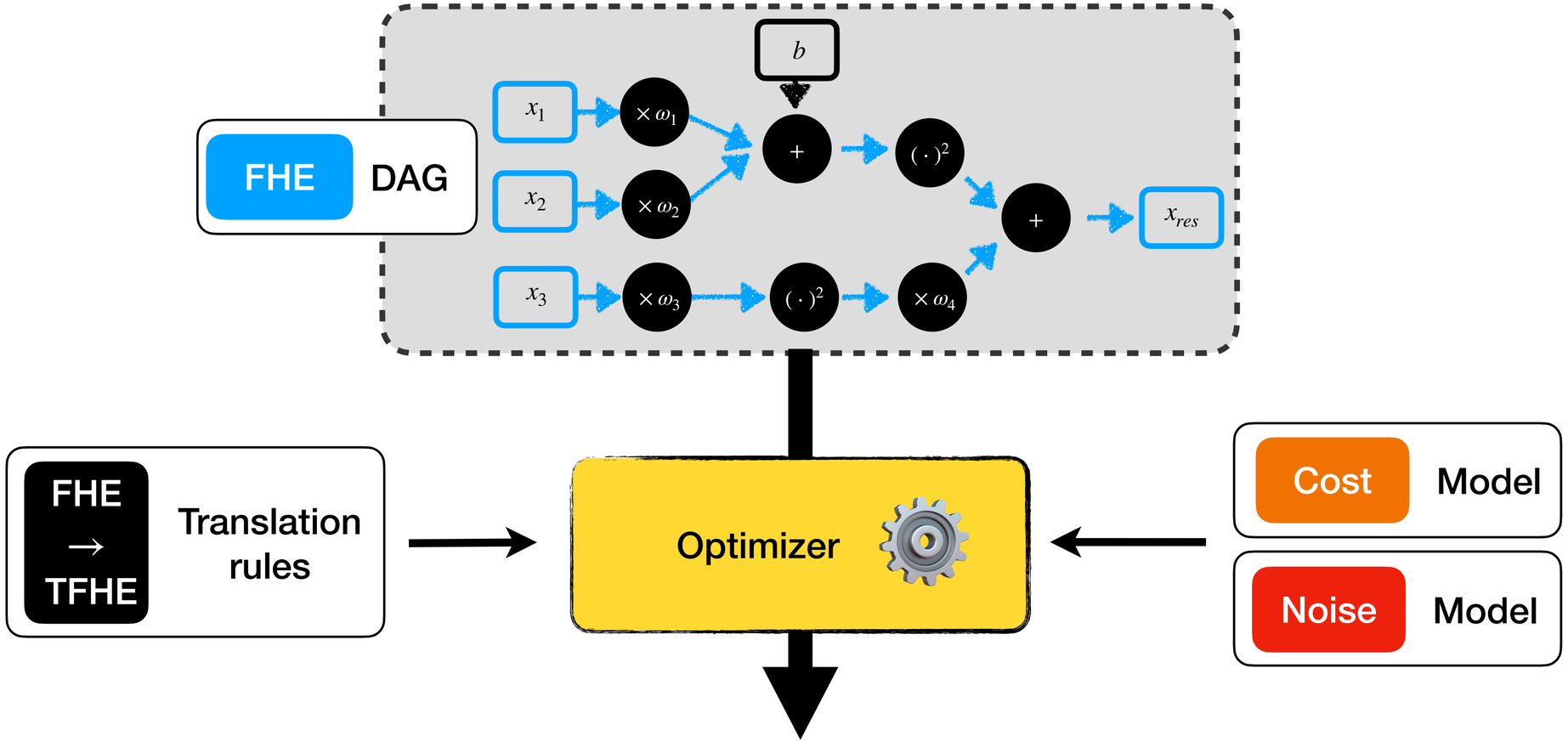
FHE → TFHE Translation rules

Cost Model

Noise Model

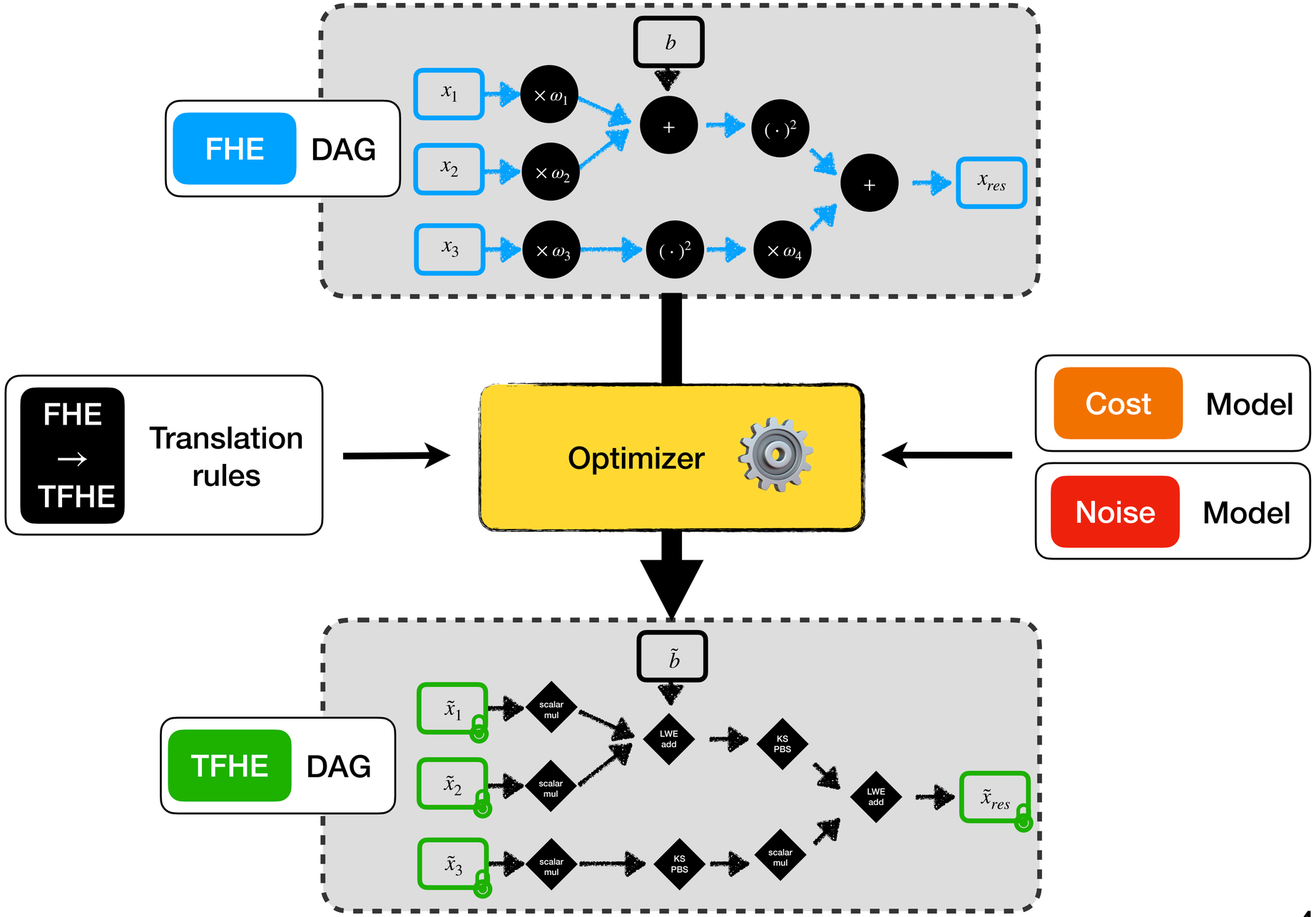
Optimizer

Overview



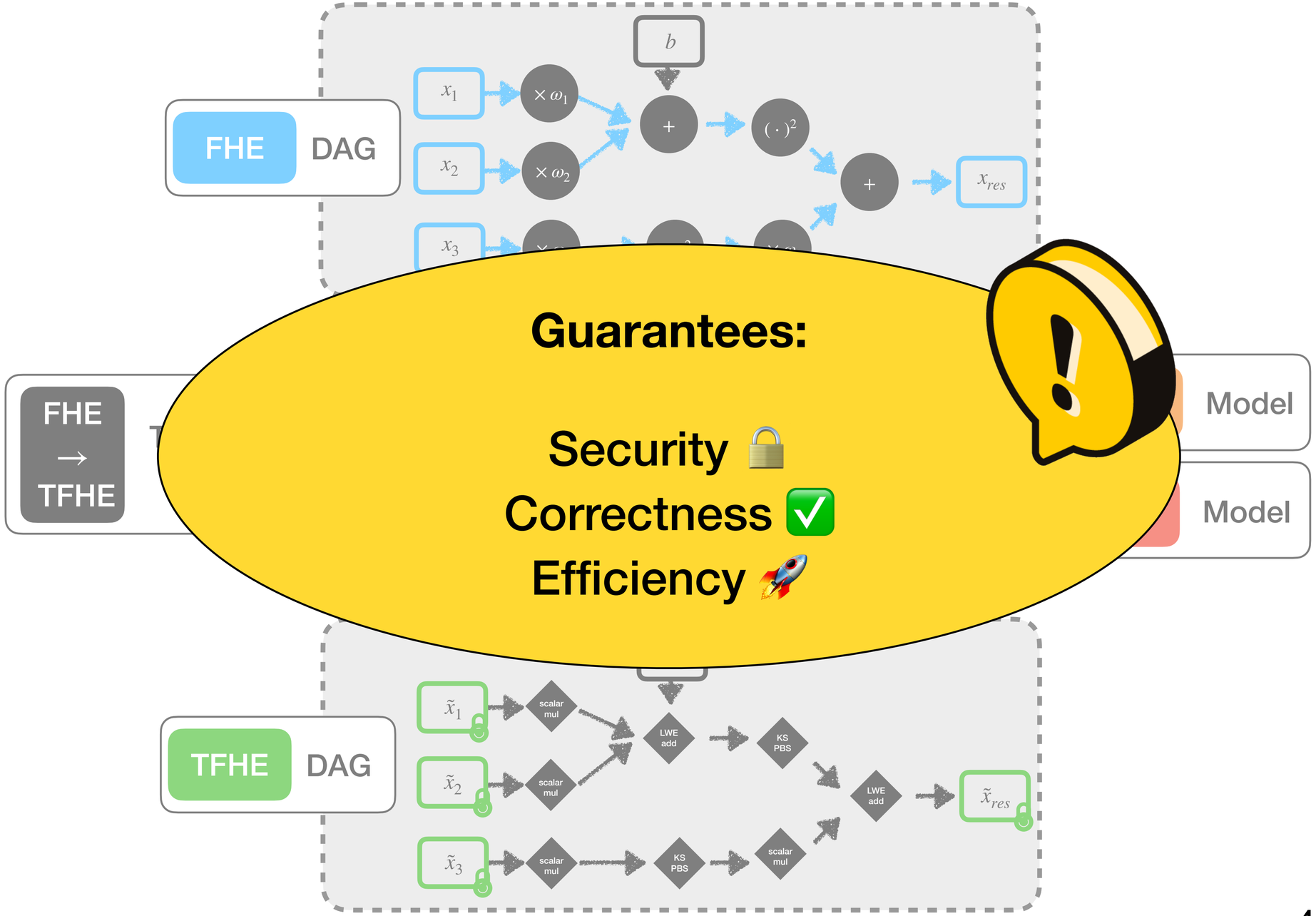
Optimizer

Overview



Optimizer

Overview



Compiler

Compiler

MLIR

High level abstraction for front ends

Compiler

MLIR

Rely on the Optimizer

High level abstraction for front ends

Compiler

MLIR

Rely on the Optimizer

Hardware optimization, distribution…

High level abstraction for front ends

Compiler

MLIR

Rely on the Optimizer

Hardware optimization, distribution…

High level abstraction for front ends

Built on top of MLIR

Compiler

MLIR

Rely on the Optimizer

Hardware optimization, distribution…

High level abstraction for front ends

Built on top of MLIR

code generation and llvm toolchain as backend

standard optimization: memory footprint, buffer reuse, avoid copying, merging loops...

Compiler

MLIR

Rely on the Optimizer

Hardware optimization, distribution…

High level abstraction for front ends

Built on top of MLIR

code generation and llvm toolchain as backend

standard optimization: memory footprint, buffer reuse, avoid copying, merging loops...

```
func.func @main(%arg0: !FHE.eint<16>) -> !FHE.eint<16> {
  %0 = arith.constant 1 : i16
  %1 = "FHE.add_eint_int"(%arg0, %0): (!FHE.eint<16>, i31) -> (!FHE.eint<16>)
  return %1: !FHE.eint<16>
}
```

Frontends

Frontend

Python

Focus on users 🔍

- No FHE-expertise needed
- Can be use-case oriented
- Different languages (Rust, Python, …)
- Easy to implement new frontends

Frontend

Python

Focus on users 🔍

- No FHE-expertise needed
- Can be use-case oriented
- Different languages (Rust, Python, ...)
- Easy to implement new frontends

```python
import concrete.numpy as cnp

def add(x, y):
    return x + y

compiler = cnp.Compiler(add, {"x": "encrypted", "y": "clear"})

inputset = [(2, 3), (0, 0), (1, 6), (7, 7), (7, 1)]
circuit = compiler.compile(inputset)

x = 4
y = 4

clear_evaluation = add(x, y)
homomorphic_evaluation = circuit.encrypt_run_decrypt(x, y)

print(x, "+", y, "=", clear_evaluation, "=", homomorphic_evaluation)
```

Conclusion

Takeaway

And future work

Takeaway

- Community effort towards FHE adoption
- Still obstacles to overcome: focus on the users!
- We built the foundations of an FHE framework
- Requires many different expertises
  (hardware, FHE, optimization, security, user experience, …)

Takeaway

And future work

Takeaway

- Community effort towards FHE adoption
- Still obstacles to overcome: focus on the users!
- We built the foundations of an FHE framework
- Requires many different expertises
  (hardware, FHE, optimization, security, user experience, …)

Future Work

- Improvement of Concrete framework
  (contributions and suggestions are more than welcome!)
- Research efforts on FHE, optimization, …
- Upcoming FHE dedicated pieces of hardware
- Teach developers

Thank you!

🙏

https://github.com/zama-ai/concrete

www.zama.ai