HELIOPOLIS: Verifiable Computation over Homomorphically Encrypted Data from Interactive Oracle Proofs is Practical

Diego F. Aranha, Anamaria Costache, **Antonio Guimarães**, Eduardo Soria-Vazquez

AARHUS UNIVERSITY

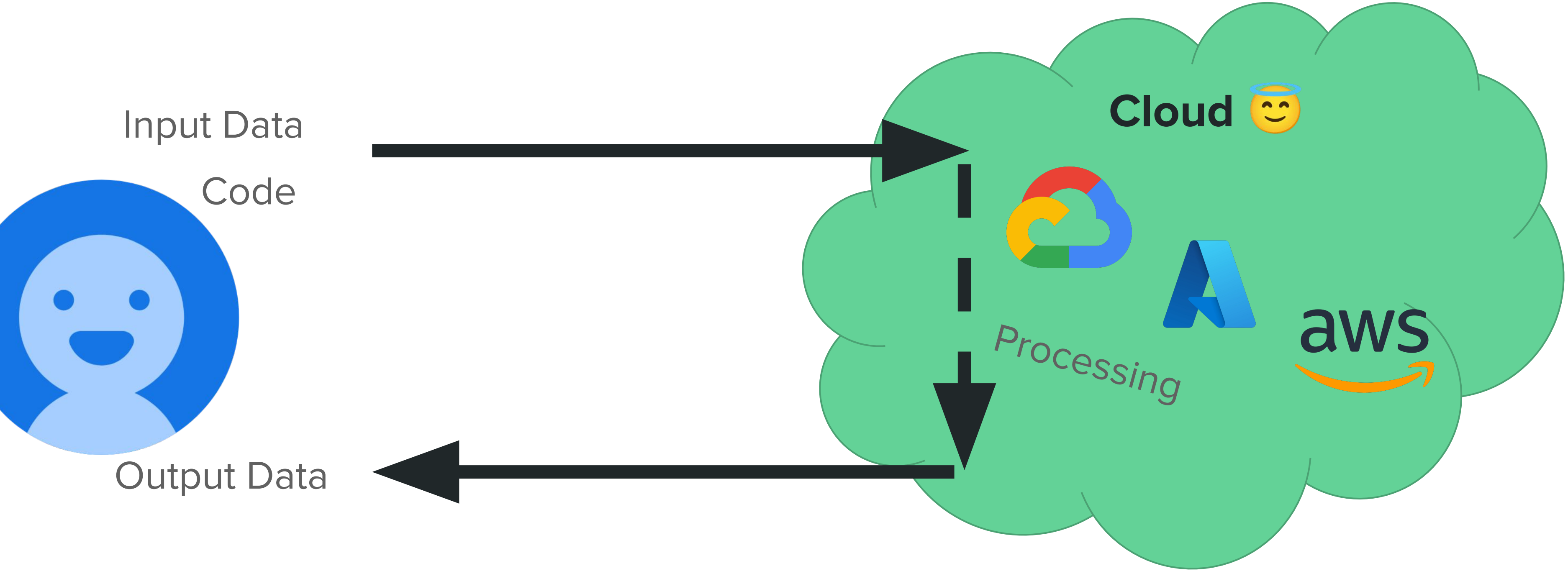
NTNU
Norwegian University of Science and Technology

institute IMDEA software

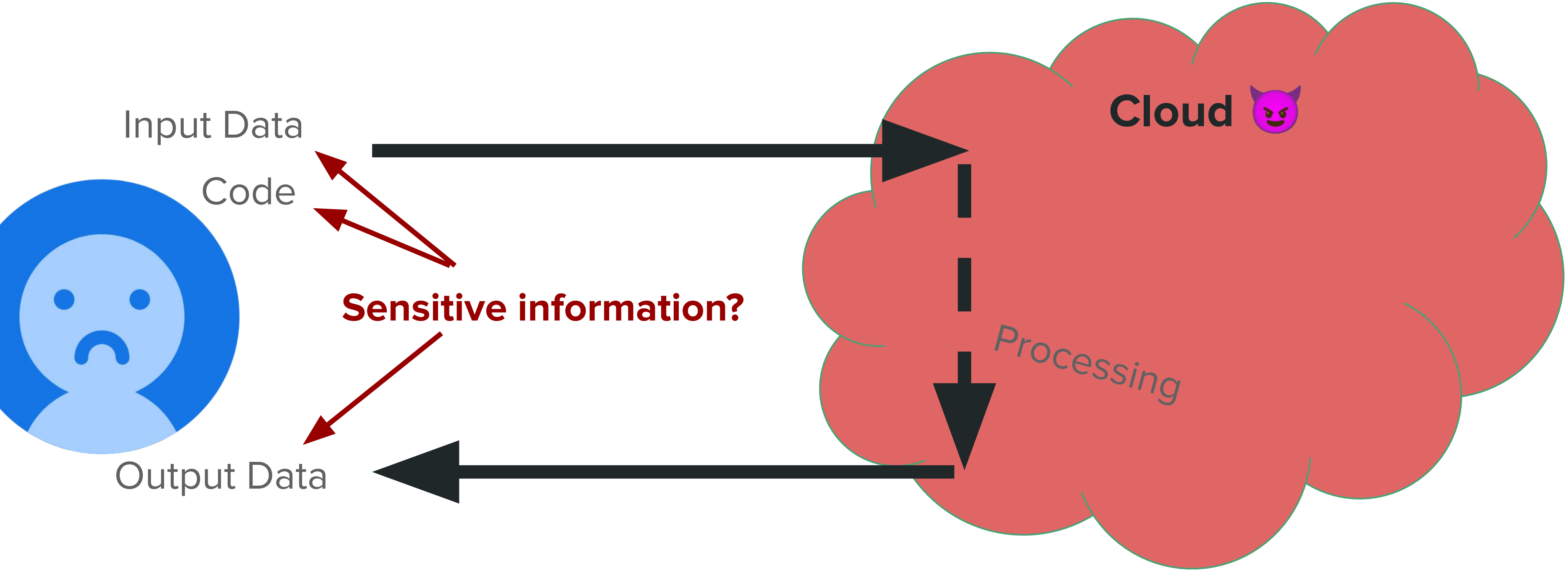
TII Technology Innovation Institute

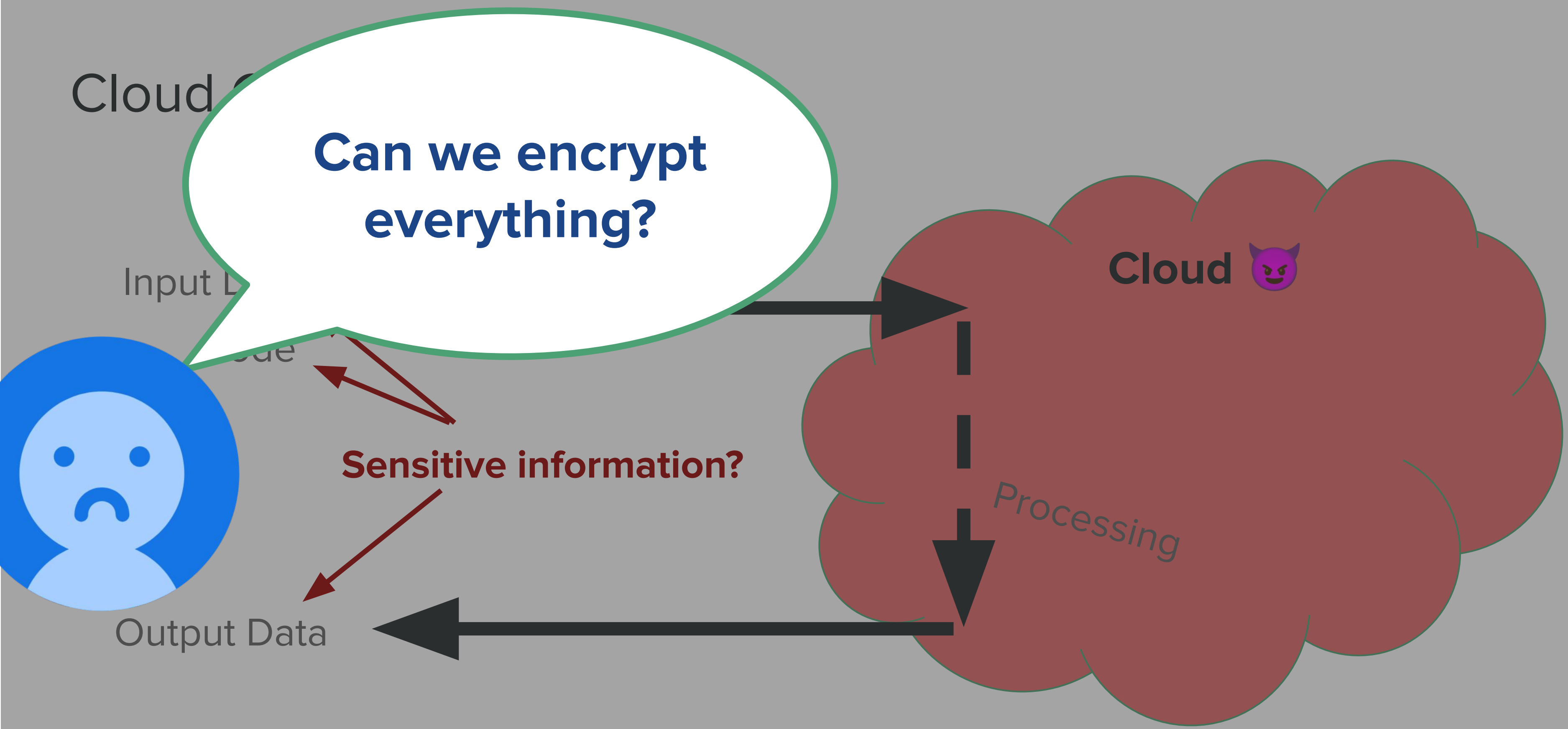
Context

Cloud Computing

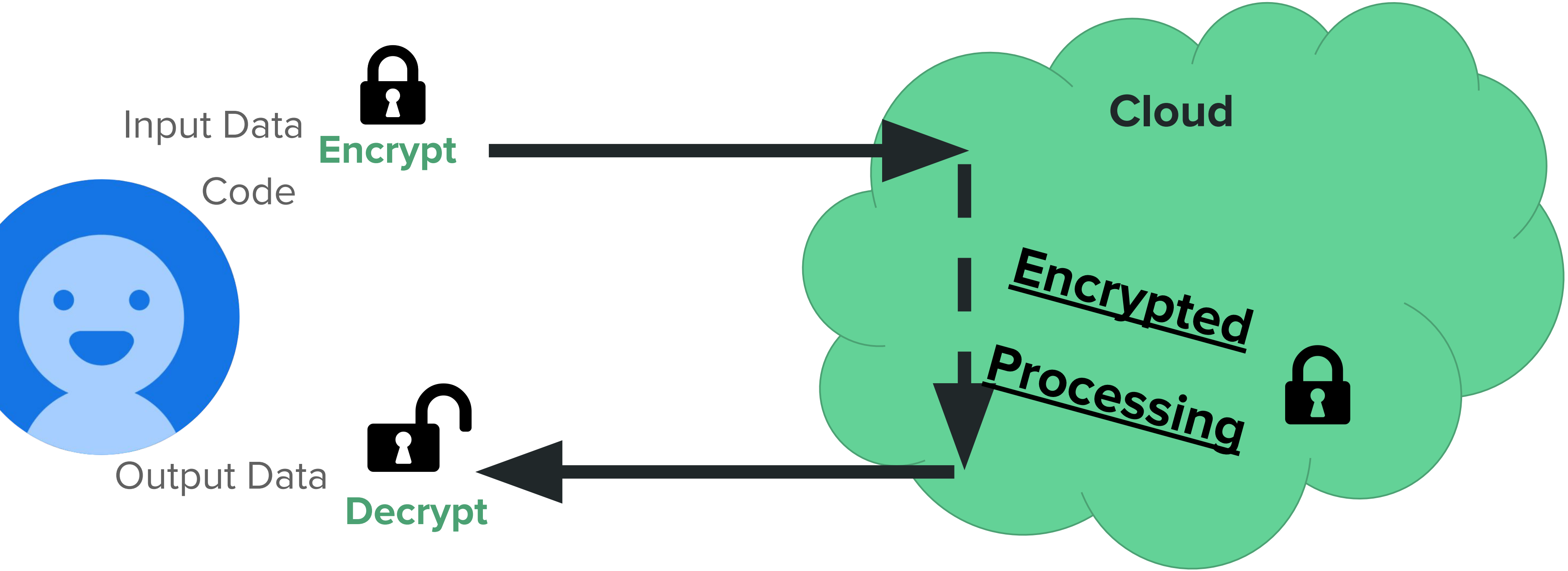


Cloud Computing



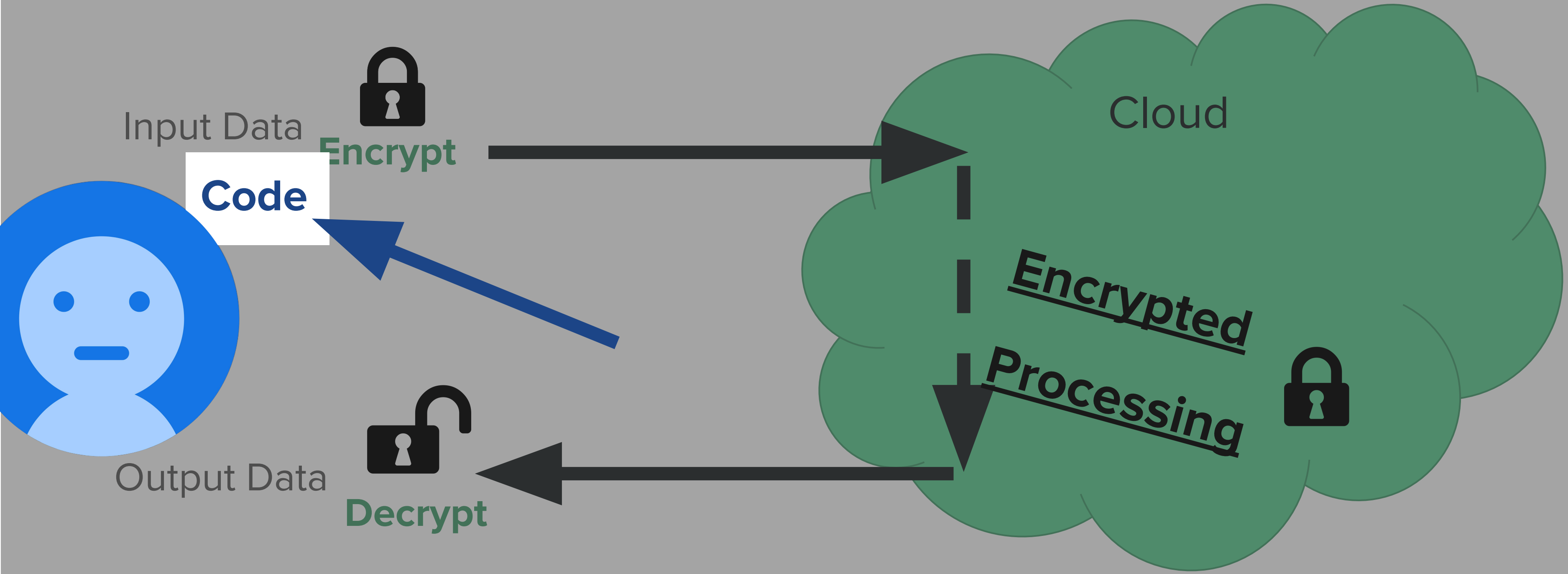

Cloud C
Can we encrypt everything?
Input D
Code
Sensitive information?
Output Data
Cloud 😈
Processing

Homomorphic Encryption (HE)



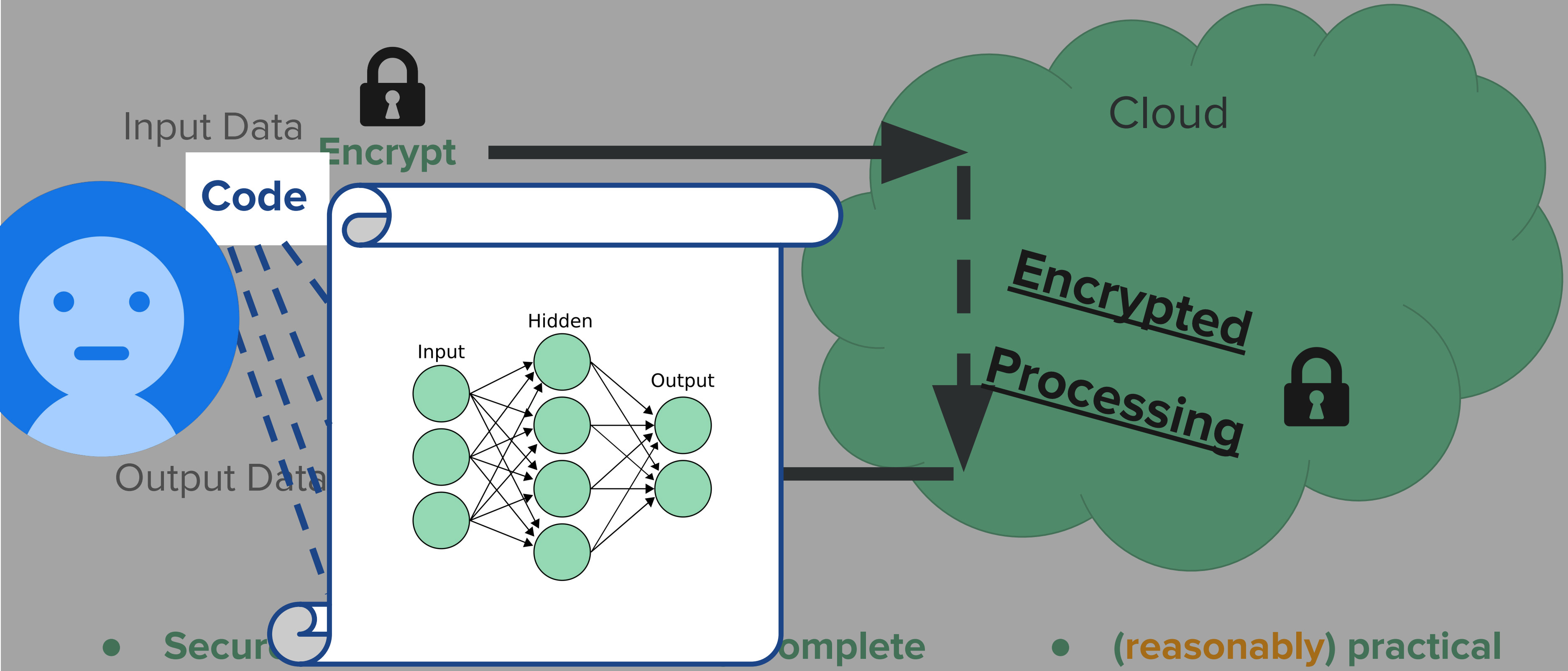
- **Secure**
- **Functionally Complete**
- **(reasonably) practical**

Homomorphic Encryption (HE)



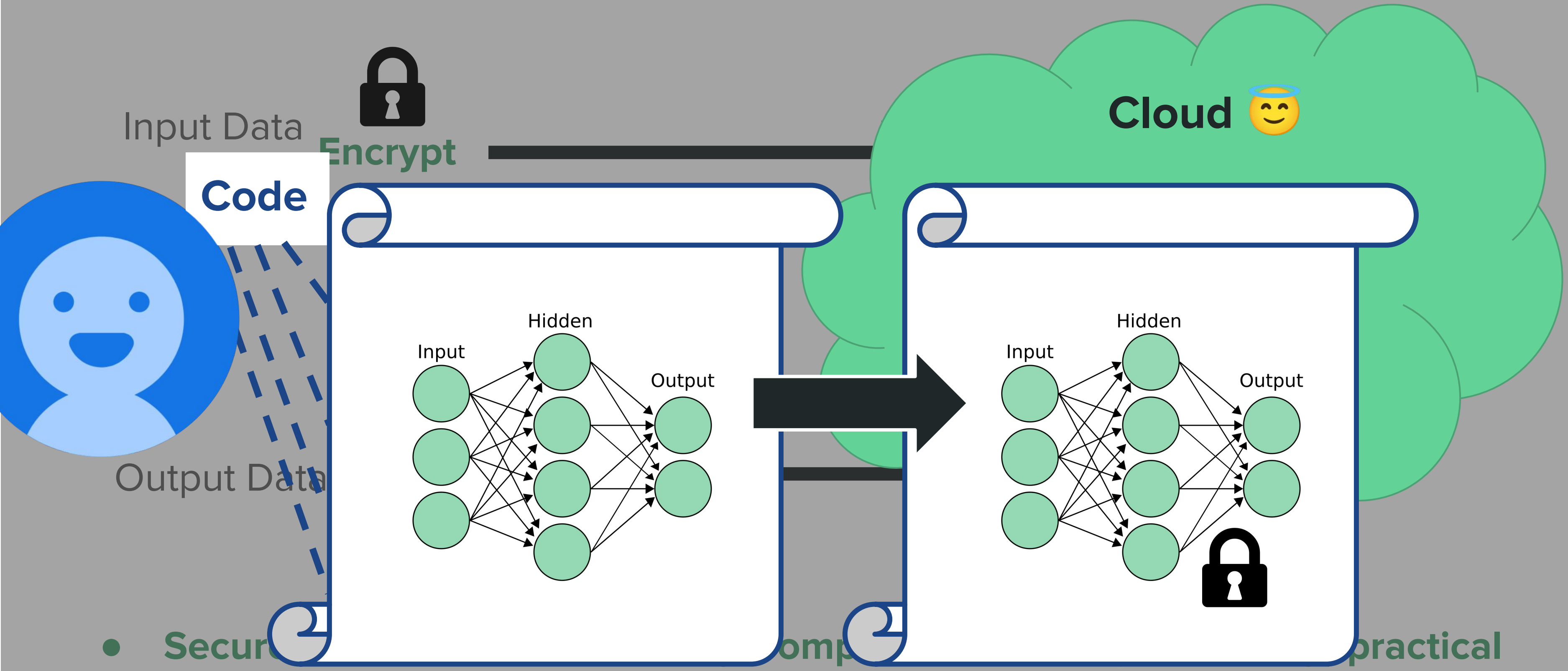
- **Secure**
- **Functionally Complete**
- **(reasonably) practical**

Homomorphic Encryption (HE)



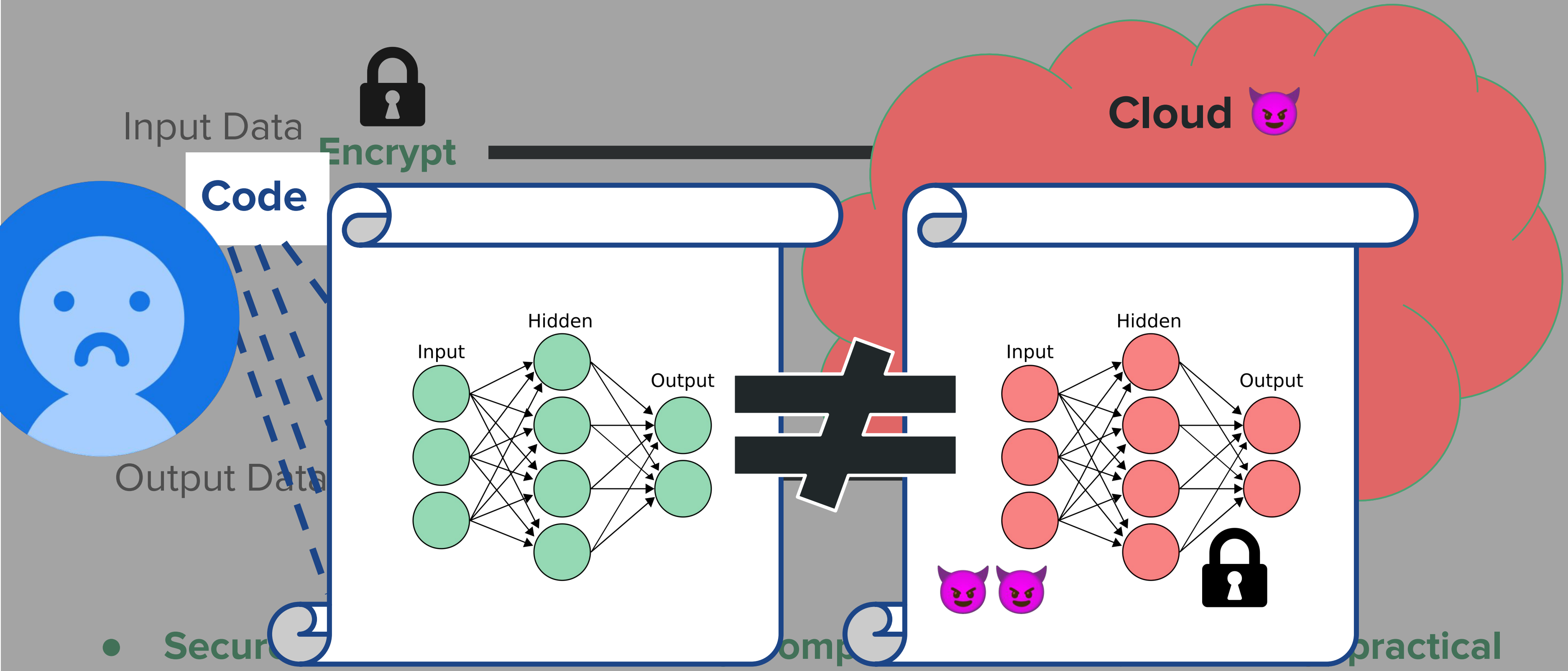
- Secur… …omplete
- (reasonably) practical

Homomorphic Encryption (HE)

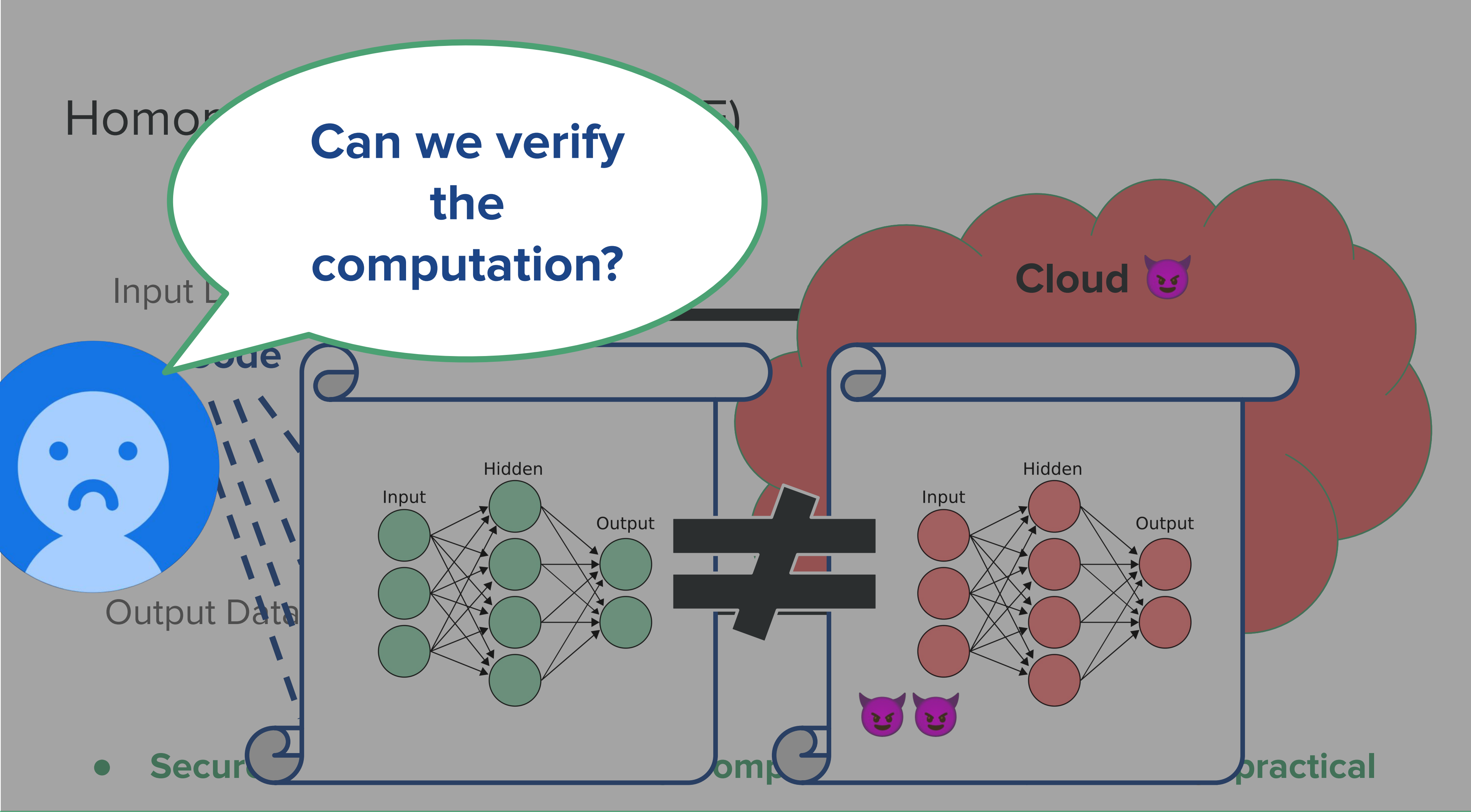


- **Secur… comp… practical**

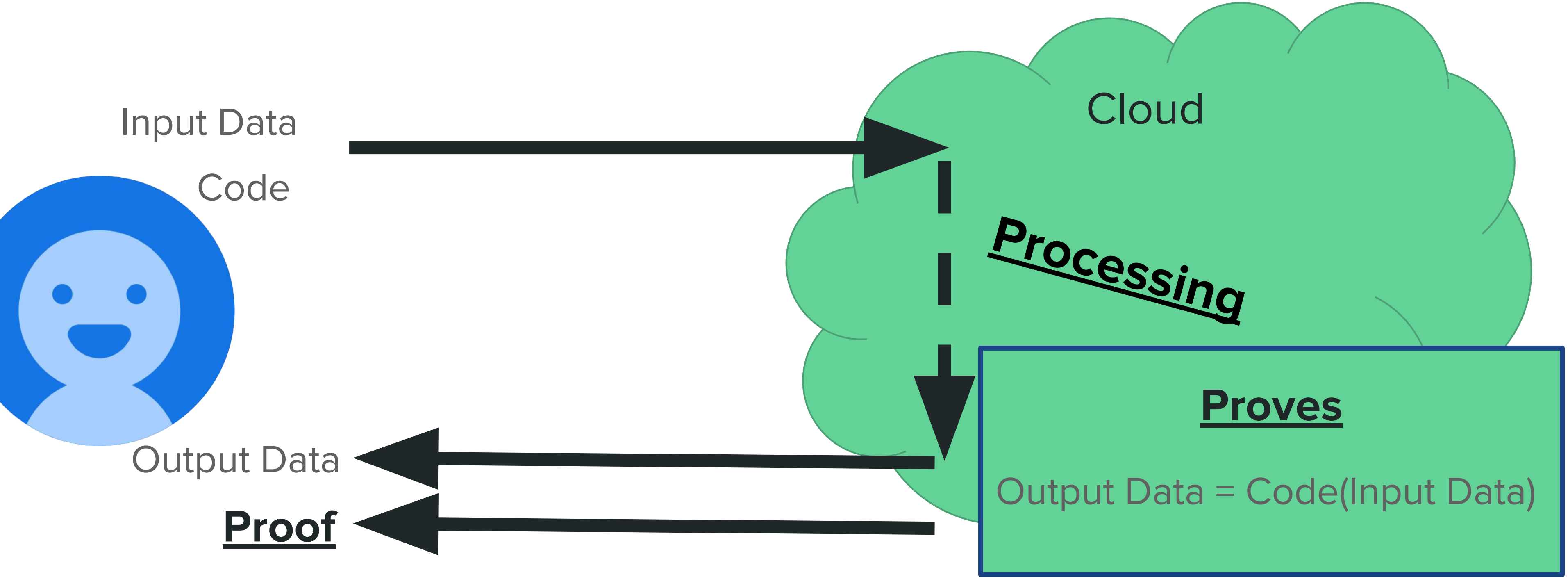
Homomorphic Encryption (HE)



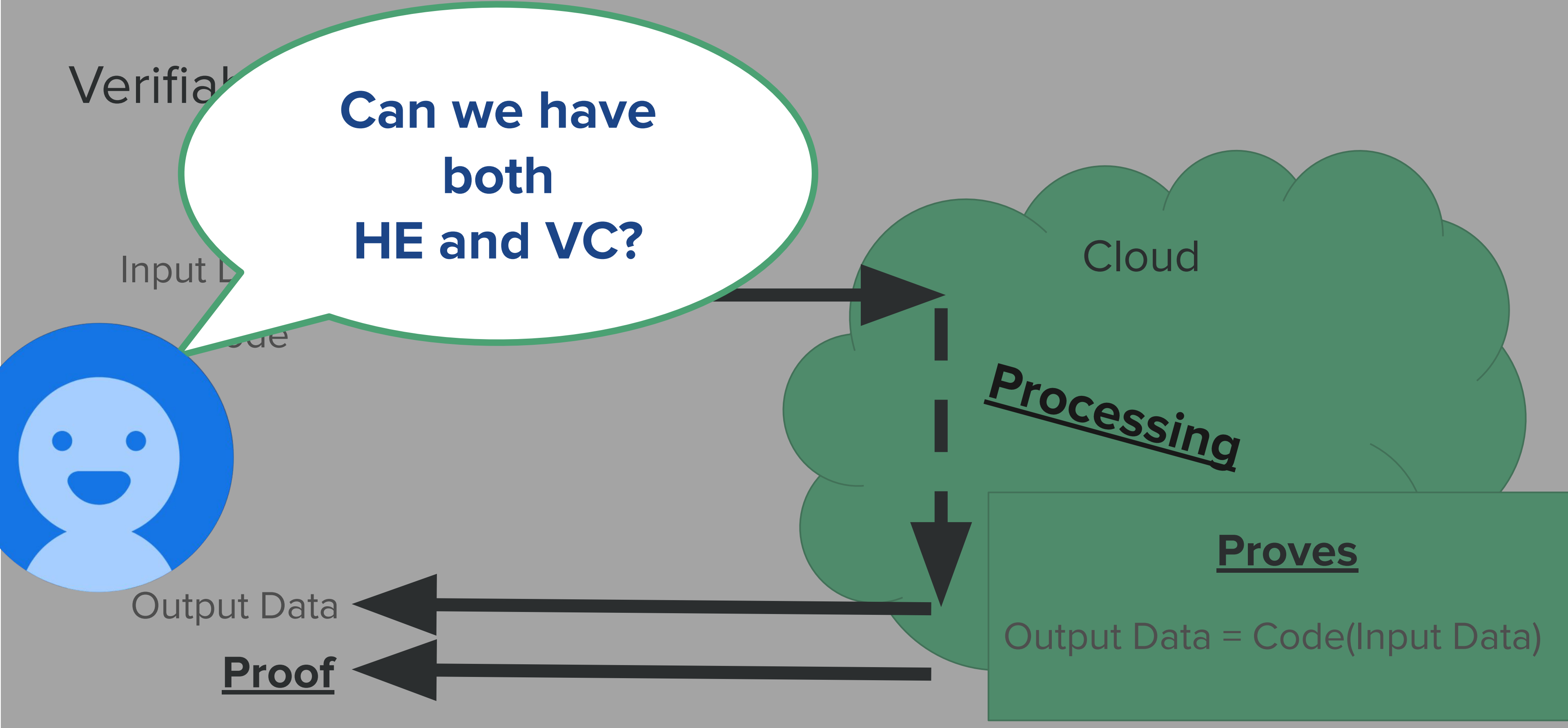
- **Secur** … **omp** … **practical**


Homo
E)
Can we verify the computation?
Input D
ode
Output Data
Cloud
Hidden
Input
Output
Hidden
Input
Output
Secur
omp
practical

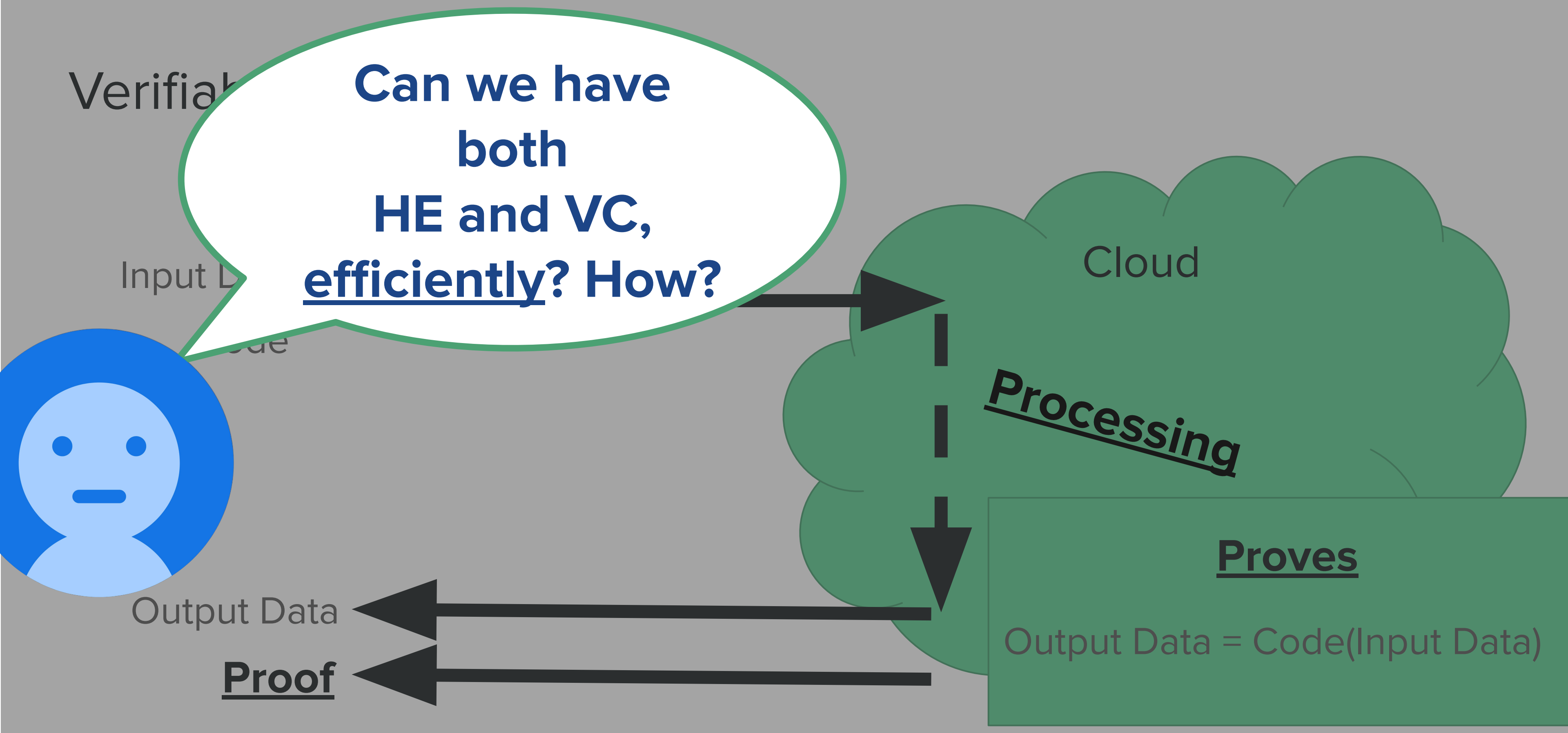
Verifiable computation (VC)



- **Sound**
- **Functionally Complete**
- **(reasonably) practical**



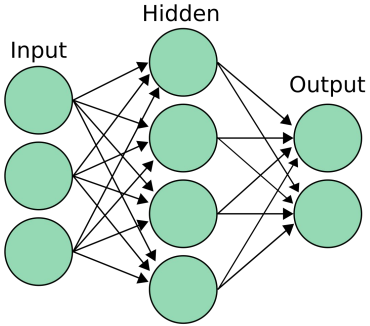
- **Sound**
- **Functionally Complete**
- **(reasonably) practical**



- **Sound**
- **Functionally Complete**
- **(reasonably) practical**

Main VC-HE approach so far

Main VC-HE approach so far

`Prove( HE(`  `) )`

⚠️ **<u>Problem:</u> VC and HE are not friendly**

Verifiable Computation

Efficient if working with:

- Fields
- Algebraic operations

Homomorphic Encryption

Efficient if working with*:

- Huge rings with composite moduli
- Rounding and modular reductions

* considering ciphertext operations

Cleartext operation

A = 35 B = 62

A*B = 2170

1. Linear, algebraic operation
2. Easy to embed in a Field
3. Takes 2 bytes of memory
4. Takes picoseconds

Homomorphic Operation

A = Encrypt(35) B = Encrypt(62)

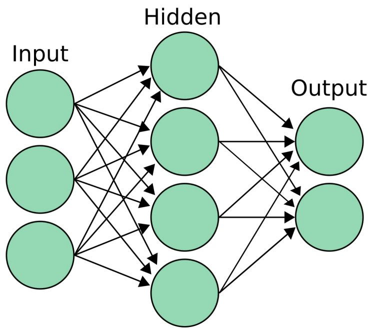
A*B = Mod-Switching(

Key-Switching(

Tensor_Multiplication(A,B)))

1. Not algebraic
2. Efficiency requires amortization
3. Takes kilobytes of memory
4. Takes microseconds

VC-HE so far

Prove(HE(

))

⚠️ **<u>Problem:</u> VC and HE are not friendly**

VC-HE so far

Prove(

The first intuition:
Instead of proving HE,
can we HE the proof?

⚠️ **Problem:** VC and HE are not friendly

Verifiable Computation

Proof systems typically require:

- Hash functions
- Large fields

Homomorphic Encryption

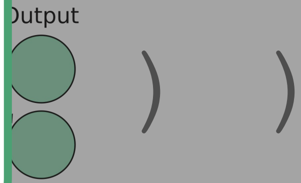
Most efficient if working with:

- Rings or small fields
- Algebraic operations

* considering plaintext operations

Our approach (HE-IOPs)

The first intuition:
Instead of proving HE,
can we HE the proof?



⚠️ **Problem:** VC and HE are not friendly

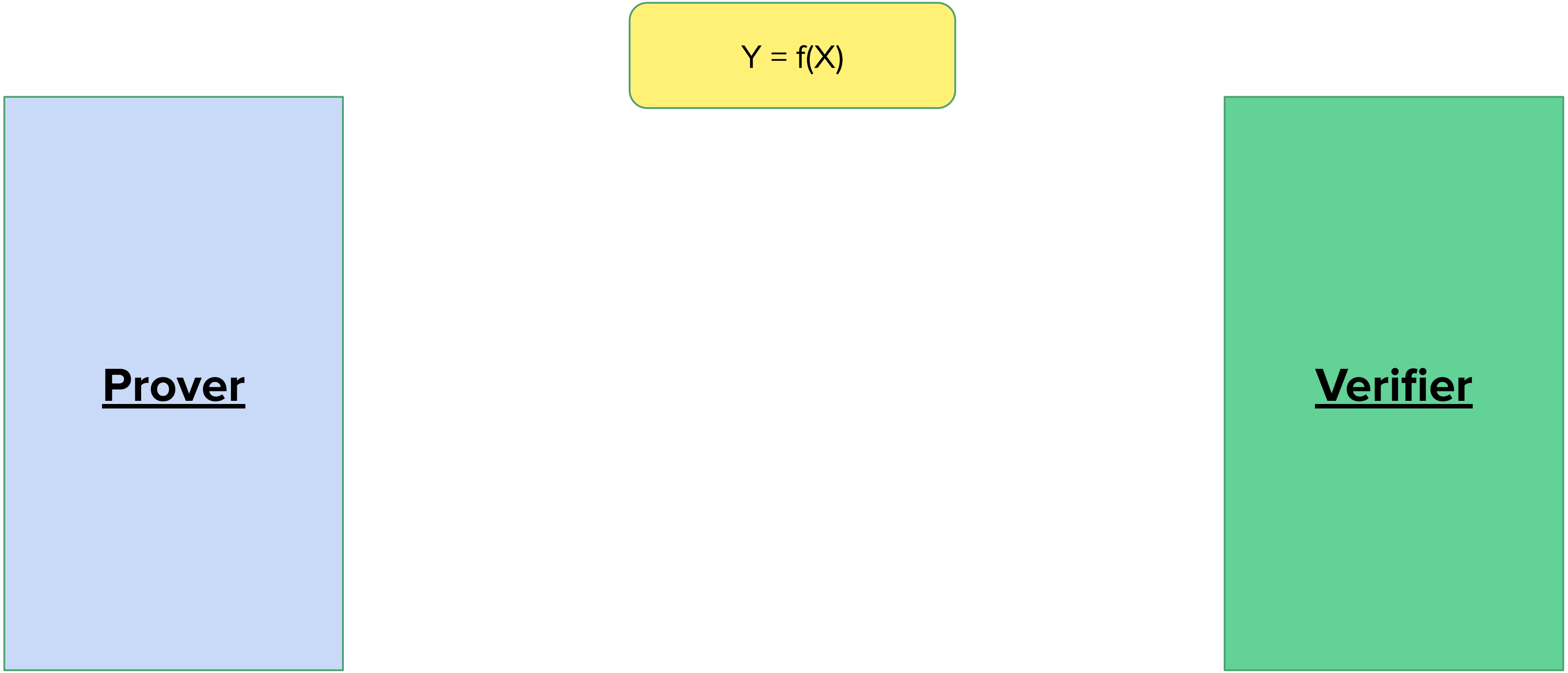
The first intuition:
Instead of proving HE,
can we HE the proof?

utput

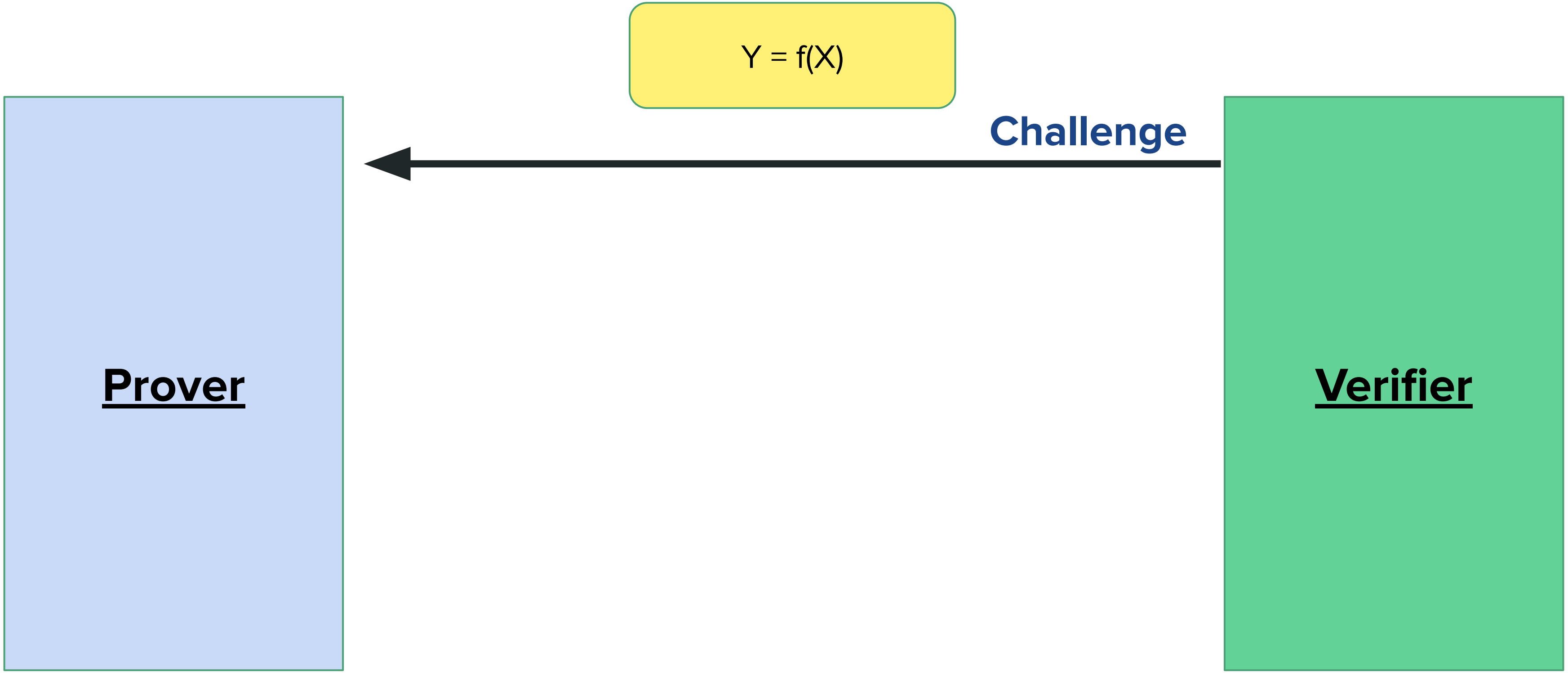
⚠️ Problem: VC and HE ar

Our method: HE the information theoretic component of the proof system

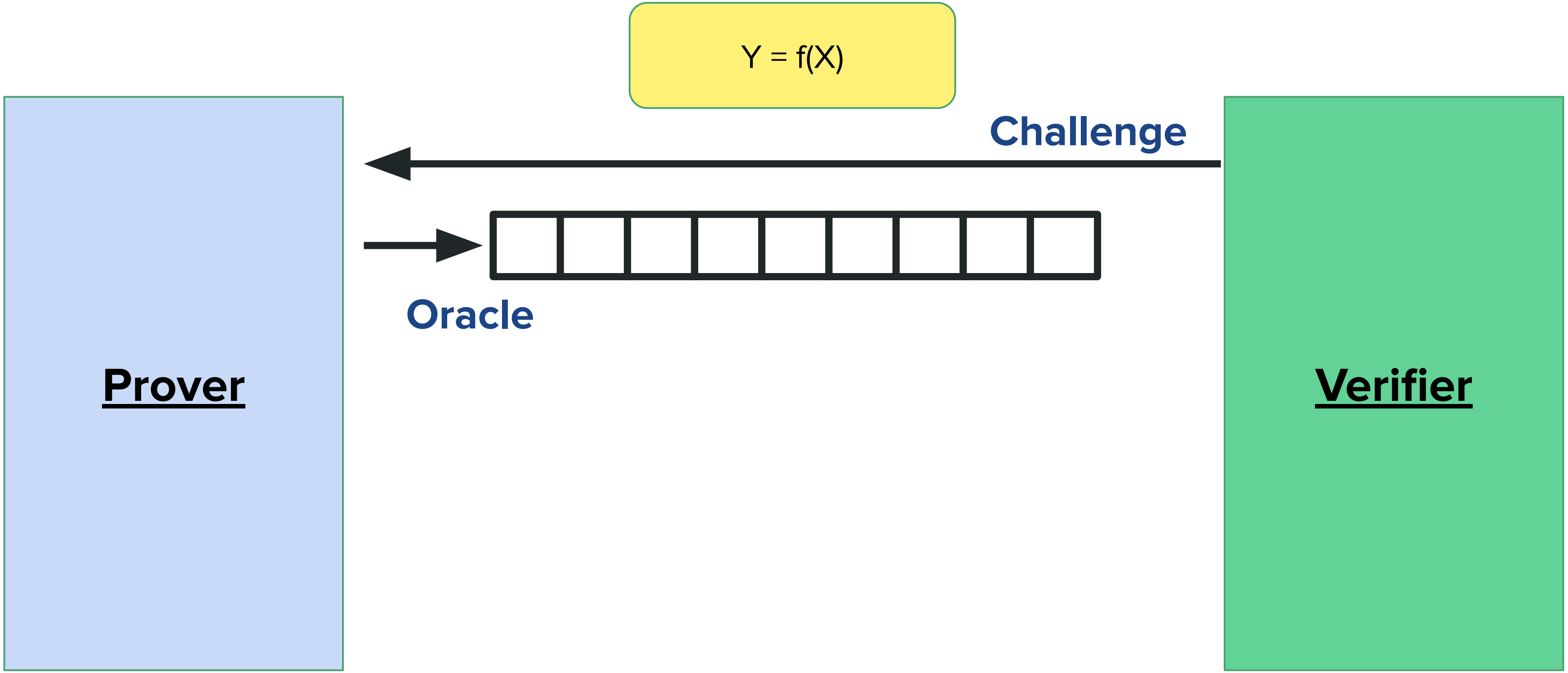
Interactive Oracle Proof (IOP)



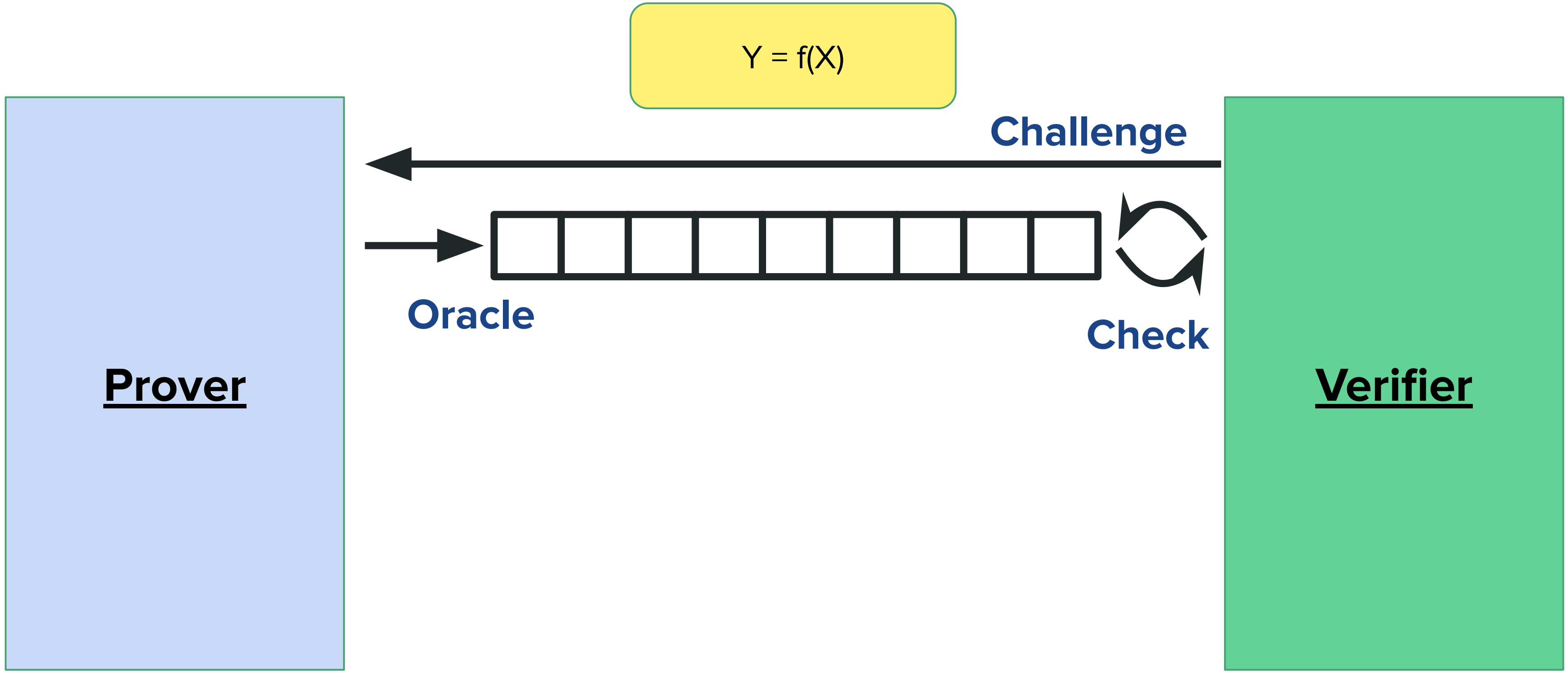
Interactive Oracle Proof (IOP)



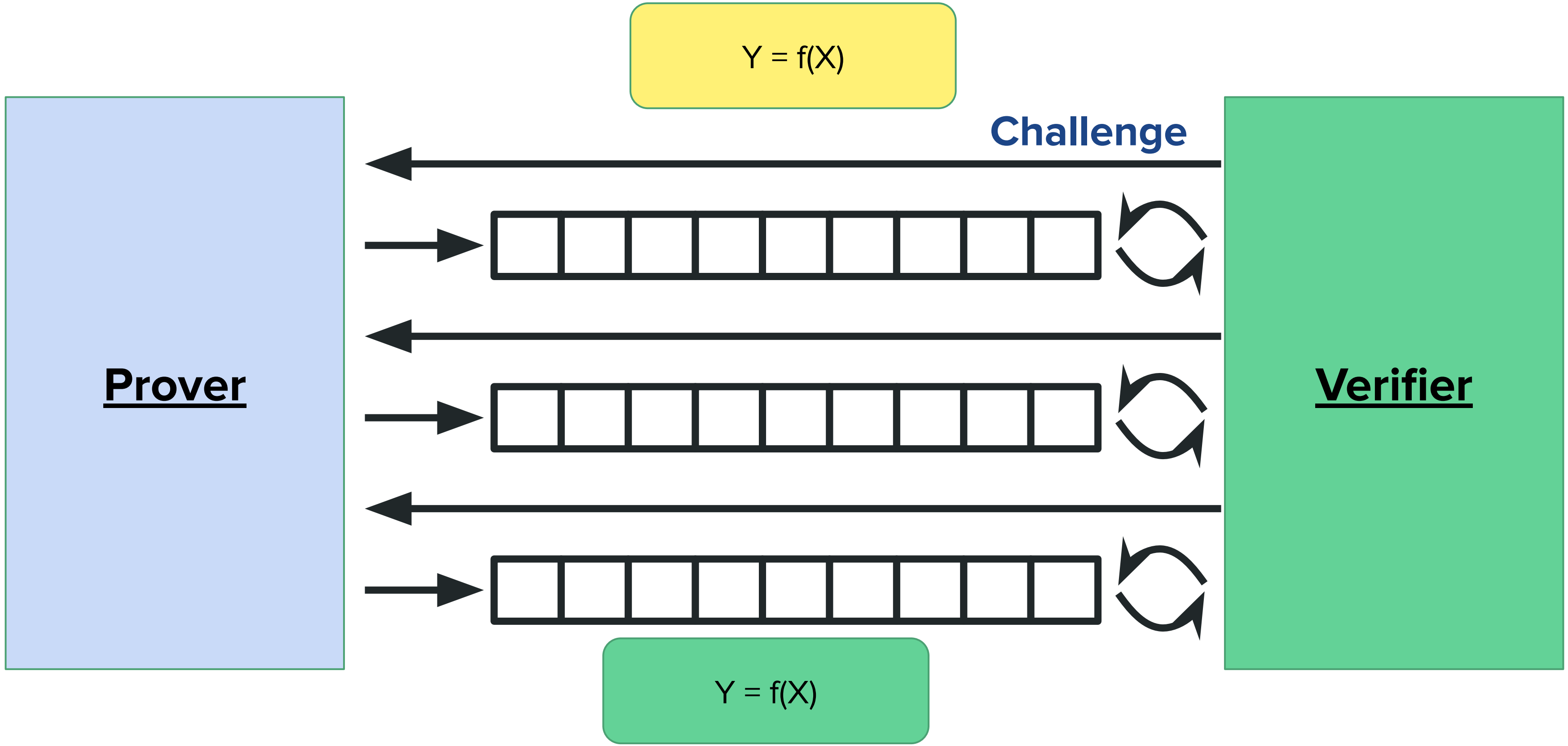
Interactive Oracle Proof (IOP)



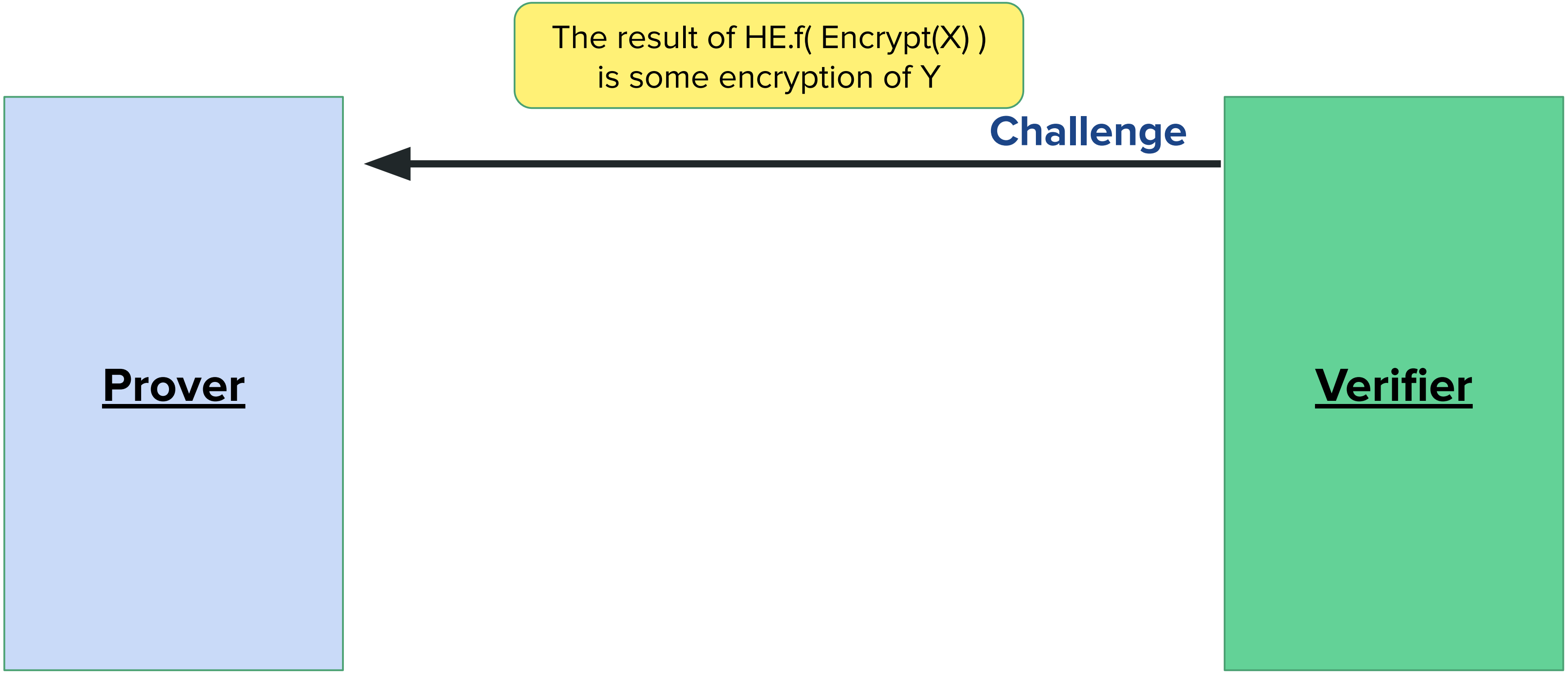
Interactive Oracle Proof (IOP)



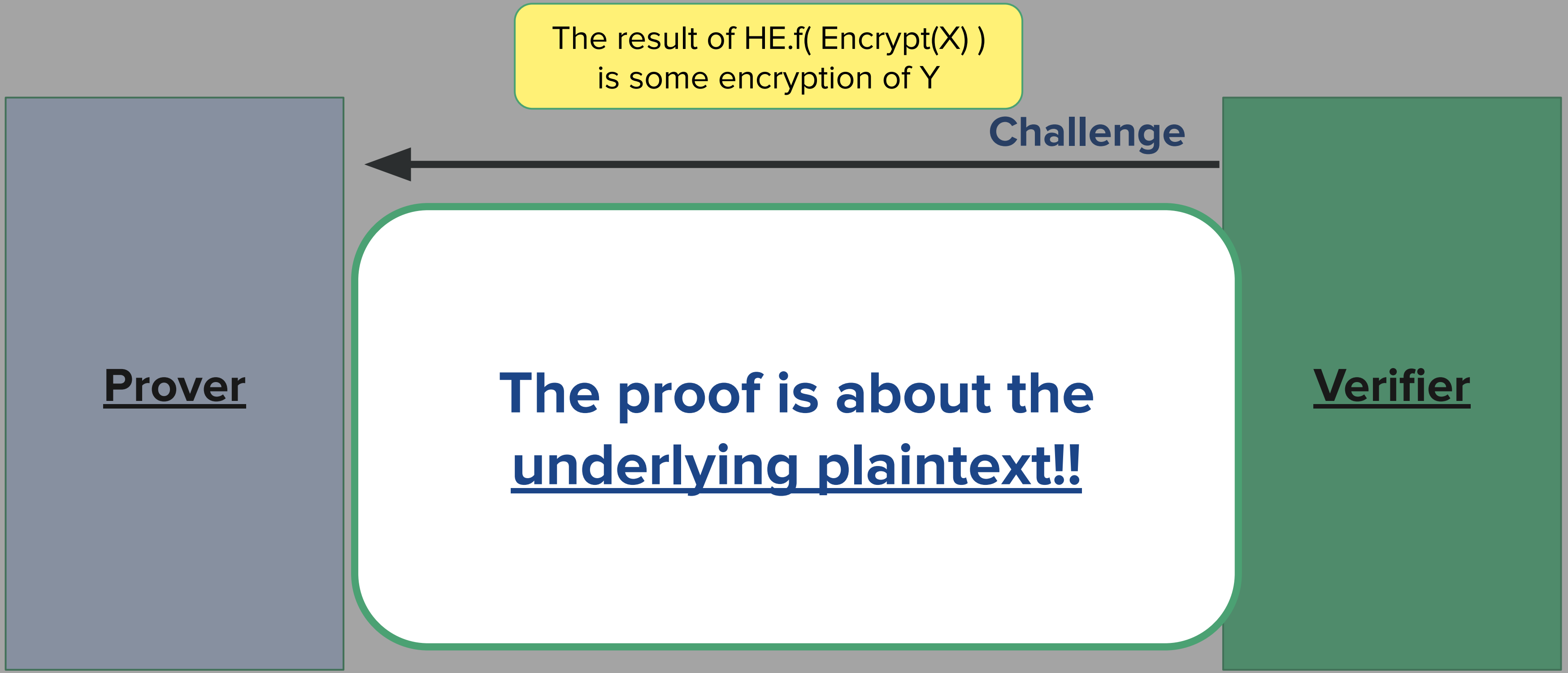
Interactive Oracle Proof (IOP)



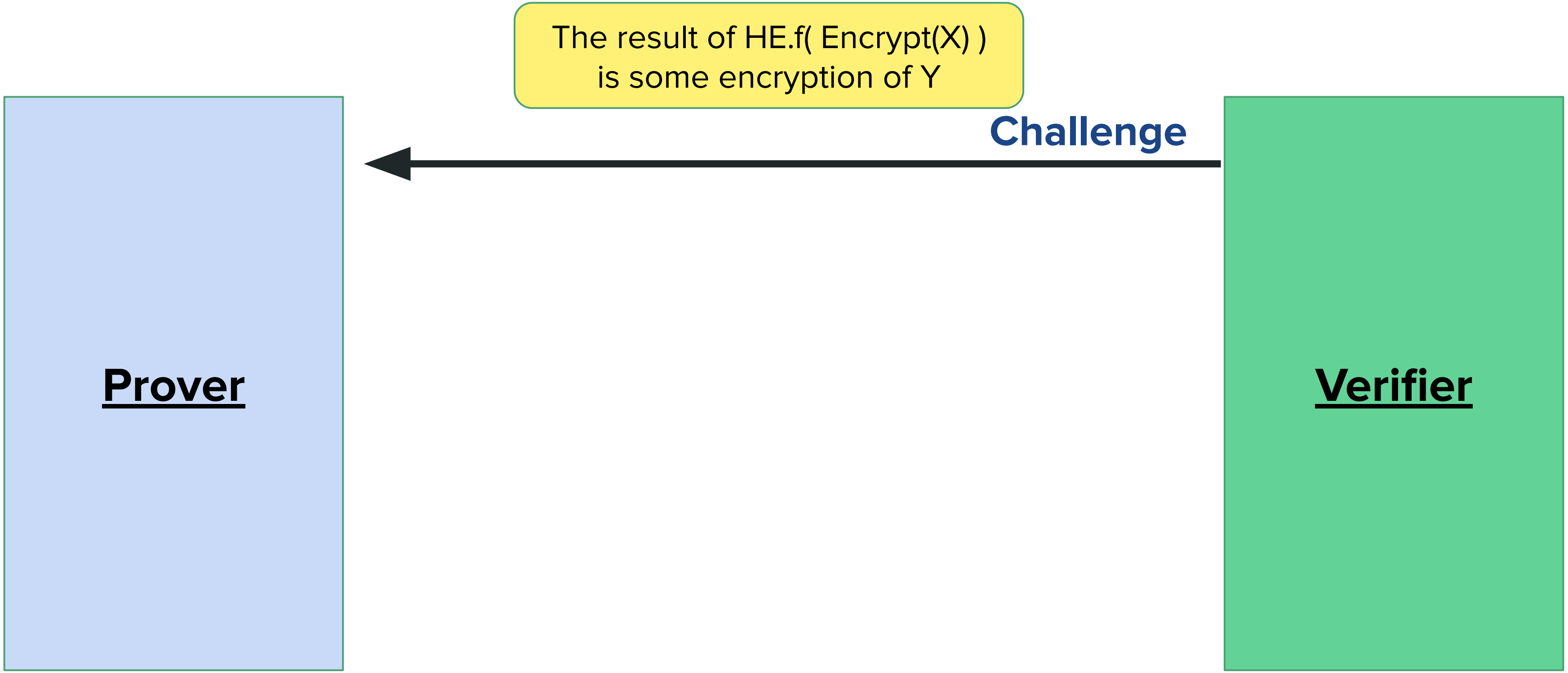
HE Interactive Oracle Proof (HE-IOP)



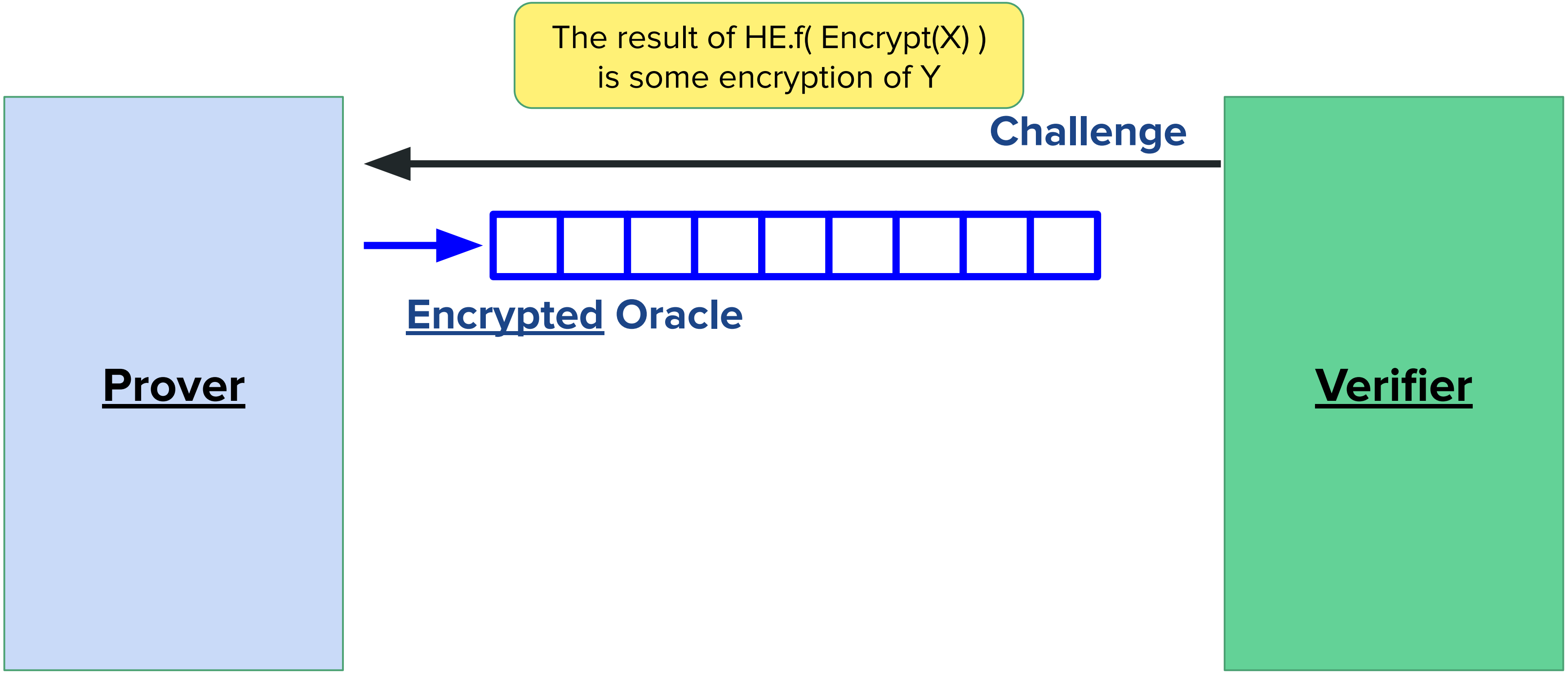
HE Interactive Oracle Proof (HE-IOP)



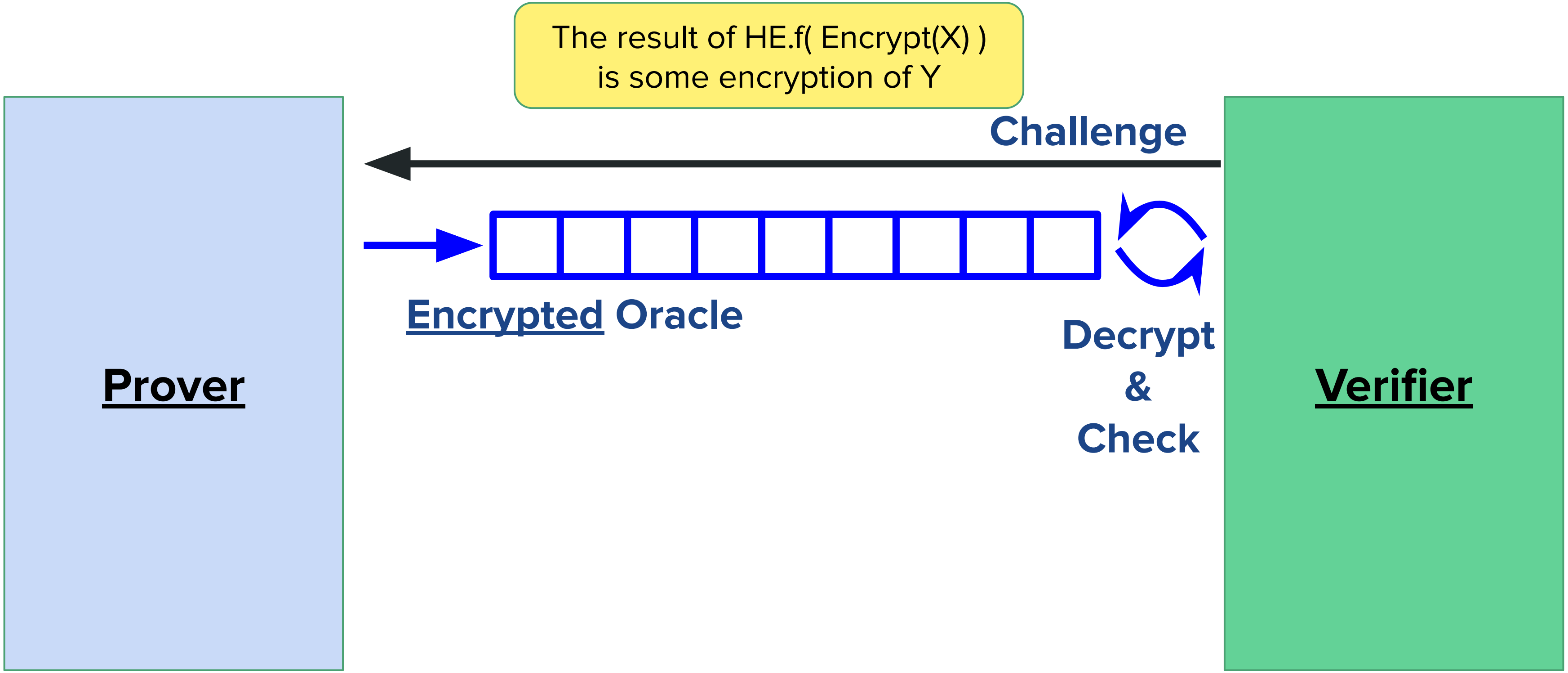
HE Interactive Oracle Proof (HE-IOP)



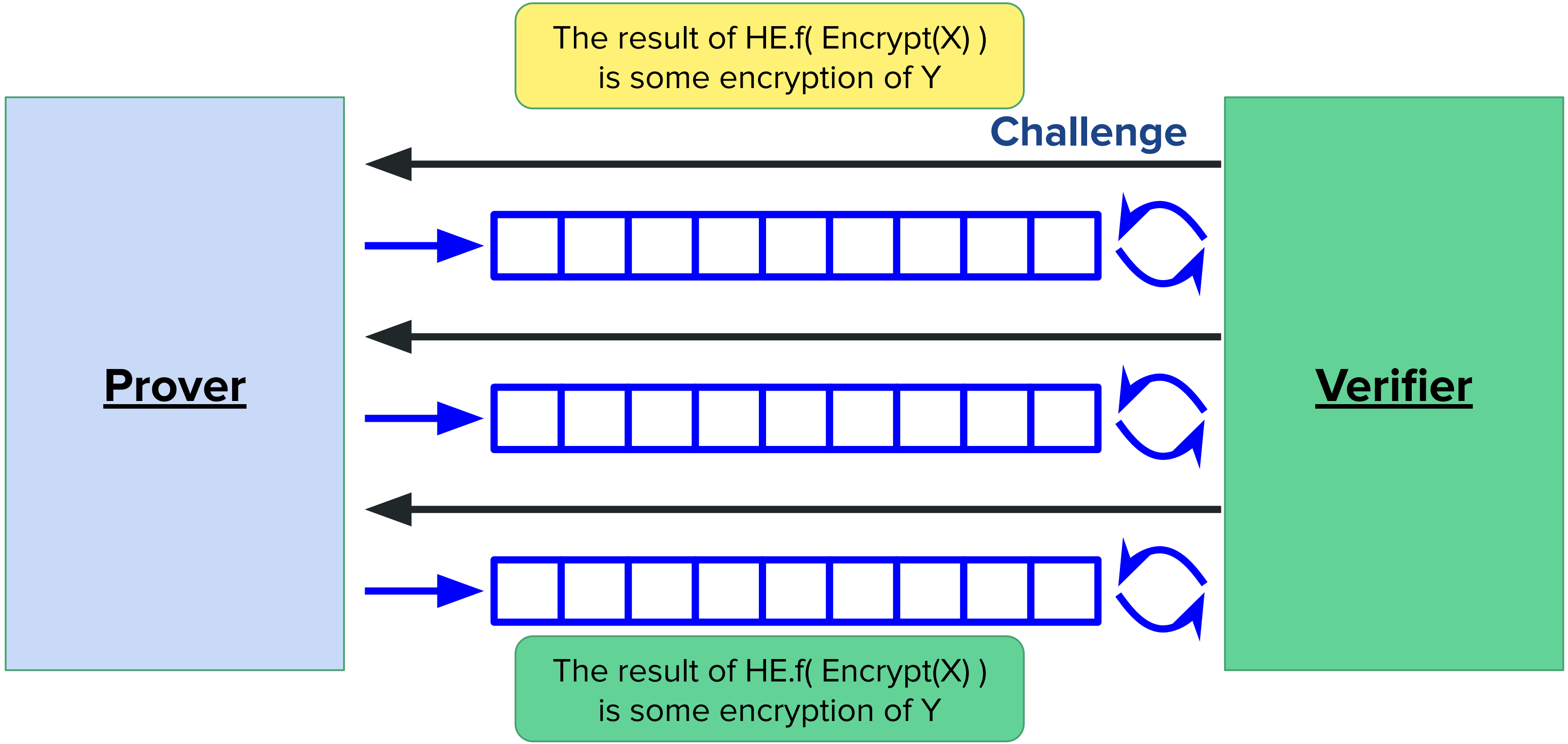
HE Interactive Oracle Proof (HE-IOP)



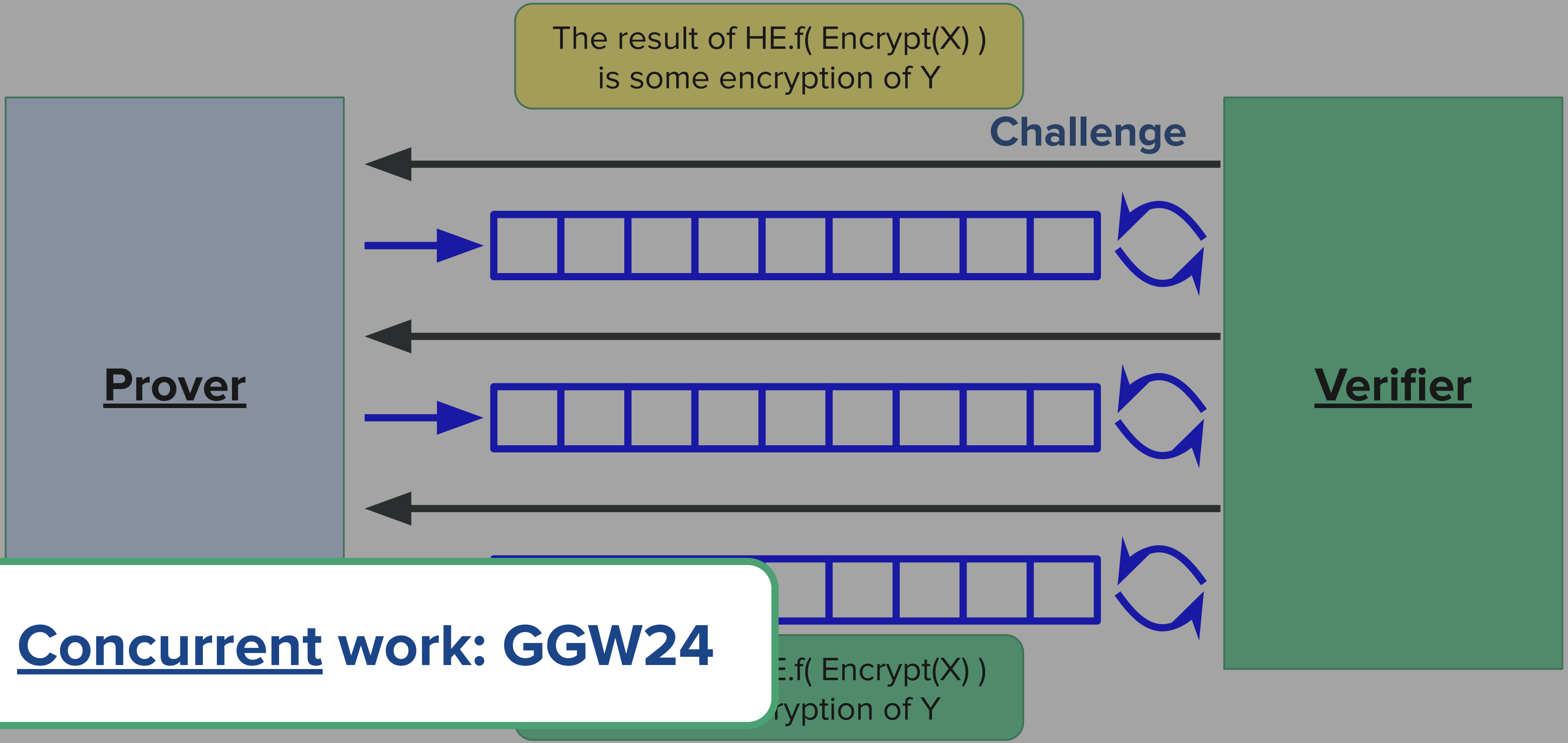
HE Interactive Oracle Proof (HE-IOP)



HE Interactive Oracle Proof (HE-IOP)



HE Interactive Oracle Proof (HE-IOP)



Concurrent work: GGW24

HE-IOPs

- We present a **generic reduction** from **HE-IOP** to the underlying **IOP**
- An **adversary** against the **HE-IOP** can be used against the underlying **IOP**
- Most parameters of the **IOP** are preserved
- We provide **zero-knowledge** (*requires circuit privacy)

HE-IOPs

- We present a … lying **IOP**
- An **adversary** … underlying **IOP**
- Most paramet…
- We provide **ze**…

Why is this better than "HE the proof"?

Verifiable Computation

Proof systems typically require:

- Hash functions
- Large fields

Homomorphic Encryption

Most efficient if working with:

- Rings or small fields
- Algebraic operations

* considering plaintext operations

Verifiable Computation

Proof systems typically require:

- ~~Hash functions~~ ✔
- Large fields ✖

Homomorphic Encryption

Most efficient if working with:

- Rings or small fields ✖
- <u>Algebraic operations</u> ✔

* considering <u>plaintext</u> operations

In practice

We implement **HE-batched-FRI**: an **HE-IOP version** of

(batched) **FRI** (**F**ast **R**eed-Solomon **I**OP of proximity)

We imple... rsion of

(batche... ximity)

HE-FRI is not only an instance of an HE-IOP!
FRI is often used to compile other IOPs!

Practical challenge 1: The field

FRI

- **Typically works with:**

$$|\mathbb{F}_p| \approx 2^{256}$$

HE-FRI

1. **Extension field:**

$$|\mathbb{F}_{p^d}| \approx 2^{256}$$

2. **Efficiently implement it with a tower of extensions:**

$$|\mathbb{F}_{p^{2^{2^{2\cdots}}}}| \approx 2^{256}$$

3. **Tensoring:**

- **Each $\mathbb{F}_{p^k}$ component in a different ciphertext**

Table 3: Practical parameters for FRI based on the maximum size of the input polynomial d.

Maximum input size $\log_2(d)$	D	p	$\log_2(p)$	$\log_2(\|\mathbb{F}_{p^D}\|)$
15	16	65537	16.0	256.0
20	11	23068673	24.5	269.1
25	9	469762049	28.8	259.3
30	7	75161927681	36.1	252.9
35	7	206158430209	37.6	263.1
40	6	6597069766657	42.6	255.5
45	5	1337006139375617	50.2	251.2

HE schemes:

BGV/BFV ✓✓ FHEW/TFHE ✓ ⚠️ CKKS ❌❌

Verifiable Computation

Proof systems typically require:

- ~~Hash functions~~ ✔
- Large fields ✔

Homomorphic Encryption

Most efficient if working with:

- Rings or small fields ✔
- Algebraic operations ✔

* considering plaintext operations

Verifiable Computation

Proof systems typically require:

- ~~Hash functio~~
- Large fields

Homomorphic Encryption

Most efficient if working with:

- ...all fields ✔
- ...perations ✔
- ...aintext operations

All problems solved?

FRI

- ~~Hash functions~~ ✔
- Large fields ✔
- Deep
- Requirements for ZK

Homomorphic Encryption

Most efficient if working with:

- Rings or small fields ✔
- Algebraic operations ✔
- Small depth
- Batched computation

* considering plaintext operations

Practical challenge 2: The depth

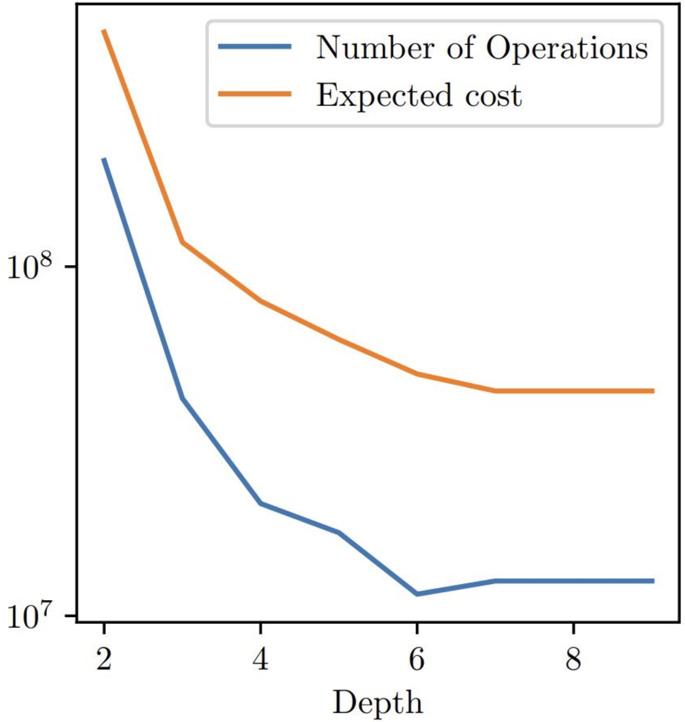
Shallow RS Encoding

- Low-depths NTTs are broadly used in HE
- **Depth:** from **$O(\log(n))$** to **2**
- **Cost:** from **$O(n \log n)$** to **$O(n\sqrt{n})$**

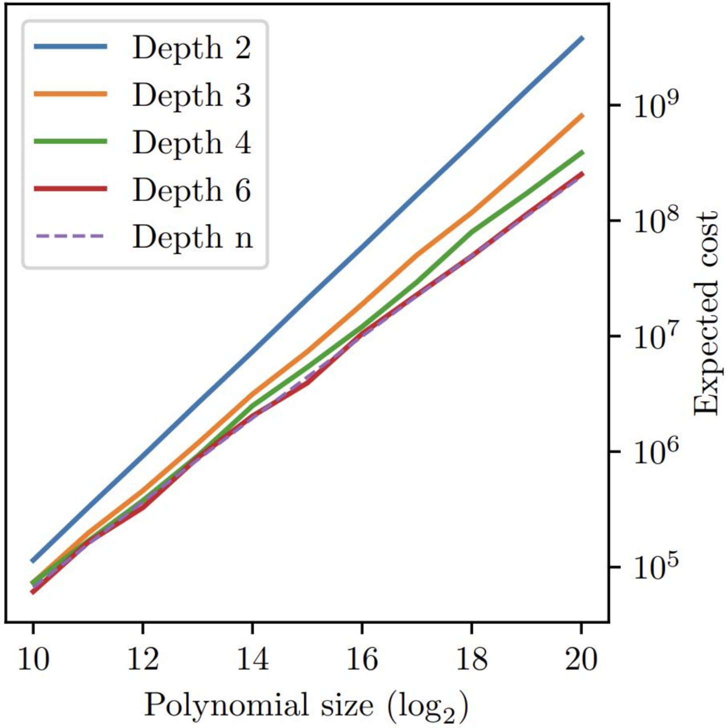
Shallow Folding

- **Does not** change overall complexity!
- **Depth:** from **$O(\log(n))$** to **1**
- **Cost:** from **$O(n)$** to **$O(n \log n)$**

Everything is **configurable**! Cost and depth are trade-offs.



(a) NTT for an input of size 2^{18}



(b) Optimal expected cost

FRI

- ~~Hash functions~~ ✔
- Large fields ✔
- Deep ✔
- Requirements for ZK

Homomorphic Encryption

Most efficient if working with:

- Rings or small fields ✔
- <u>Algebraic operations</u> ✔
- Small depth ✔
- Batched computation

* considering <u>plaintext</u> operations

Practical challenge 3: ZK and HE overhead

HE Packing

Plaintext space: $\mathcal{R}_p \mapsto \mathbb{F}_p \times \mathbb{F}_p \times \cdots \times \mathbb{F}_p$

Problem - On each check:

- The verifier **wants** to learn just **2 points** (performance)
- The prover **doesn't want** the verifier to learn more than **2 points** (ZK)
- HE packing provides at least **N = 2^{12} points**

Repack and (optionally) decompose

Parameter Set	k	N	$\log_2(q)$	Size (bytes)	Decryption Cost
$\mathfrak{P}_0$	1	512	12	8192	5120
$\mathfrak{P}_1$	2	512	25	12288	5632
$\mathfrak{P}_2$	1	1024		16384	11264
$\mathfrak{P}_3$	4	512	52	20480	6656
$\mathfrak{P}_4$	2	1024		24576	12288
$\mathfrak{P}_5$	1	2048		32768	24576

Solves HE overhead: The verifier can have HE parameters **independent** of the circuit (in practice)

HE Packing

Plaintext space: $\mathcal{R}_p \mapsto \mathbb{F}_p \times \mathbb{F}_p \times \cdots \times \mathbb{F}_p$

Problem - On each check:

- The verifier **wants** to learn just **2 points** (performance)
- The prover **doesn't want** the verifier to learn more than **2 points** (ZK)
- ~~HE packing provides at least **N = 2^{12} points**~~
- (repacked) HE packing provides **2 points**

FRI

- ~~Hash functions~~ ✔
- Large fields ✔
- Deep ✔
- Requirements for ZK ✔

Homomorphic Encryption

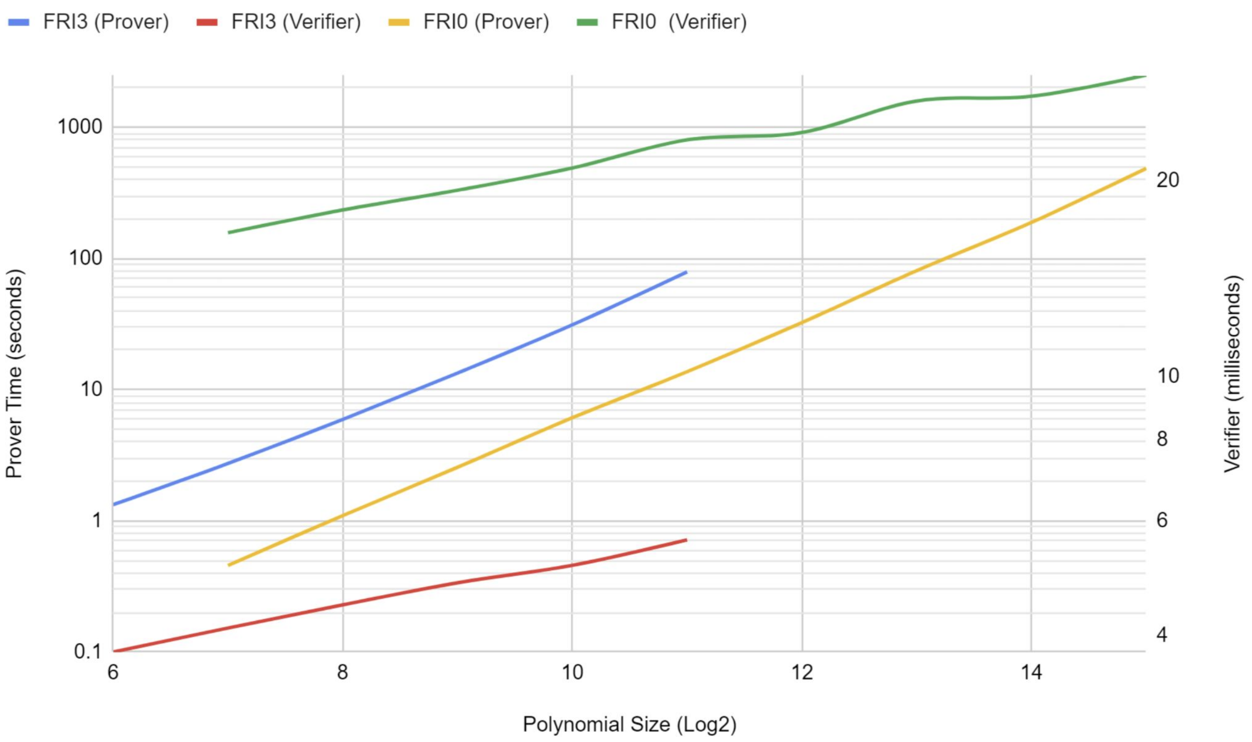
Most efficient if working with:

- Rings or small fields ✔
- Algebraic operations ✔
- Small depth ✔
- Batched computation ✔

* considering plaintext operations

Results

Results for 4096 batched polynomials



Prover: up to 32 threads - Verifier: single-threaded

For up to 2^{11}:

FRI0 (optimized for prover):

- P time: 0.2 – 5.45s
- V time: 7.08 – 12.29 ms
- Memory: 0.5 – 3.7 GB

FRI3 (optimized for verifier):

- P time: 2.74 – 78.98 s
- V time: 4.10 – 5.61 ms
- Memory: 2.0 – 23.7 GB

Implementation

- Batched for 4096 or 8192 polynomials
- **Non-interactive** (Fiat-Shamir using BLAKE3)
- **Python** with optimizations in **C/CPP**
- **Publicly available**: https://github.com/antoniocgj/HELIOPOLIS
- Artifact accepted: **IACR Results Reproduced**

https://eprint.iacr.org/2023/1949

https://github.com/antoniocgj/HELIOPOLIS

Thank you!

AARHUS UNIVERSITY

NTNU Norwegian University of Science and Technology

institute IMDEA software

TII Technology Innovation Institute

We would like to thank Zvika Brakerski for comments about our repacking optimization for the HE-Batched-FRI protocol. We also want to thank Alexander R. Block, Albert Garreta, Jonathan Katz, Justin Thaler, Pratyush Ranjan Tiwari and Michał Zajac for a useful conversation about their work [BGK+23] and confirming that their analysis does not require finite fields to be prime. This work was partly done while A. Guimarães was a Ph.D. student at University of Campinas, Brazil. He was supported by the São Paulo Research Foundation under grants 2013/082937, 2019/12783-6, and 2021/09849-5. This work is partially funded by the European Union (GA 101096435). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

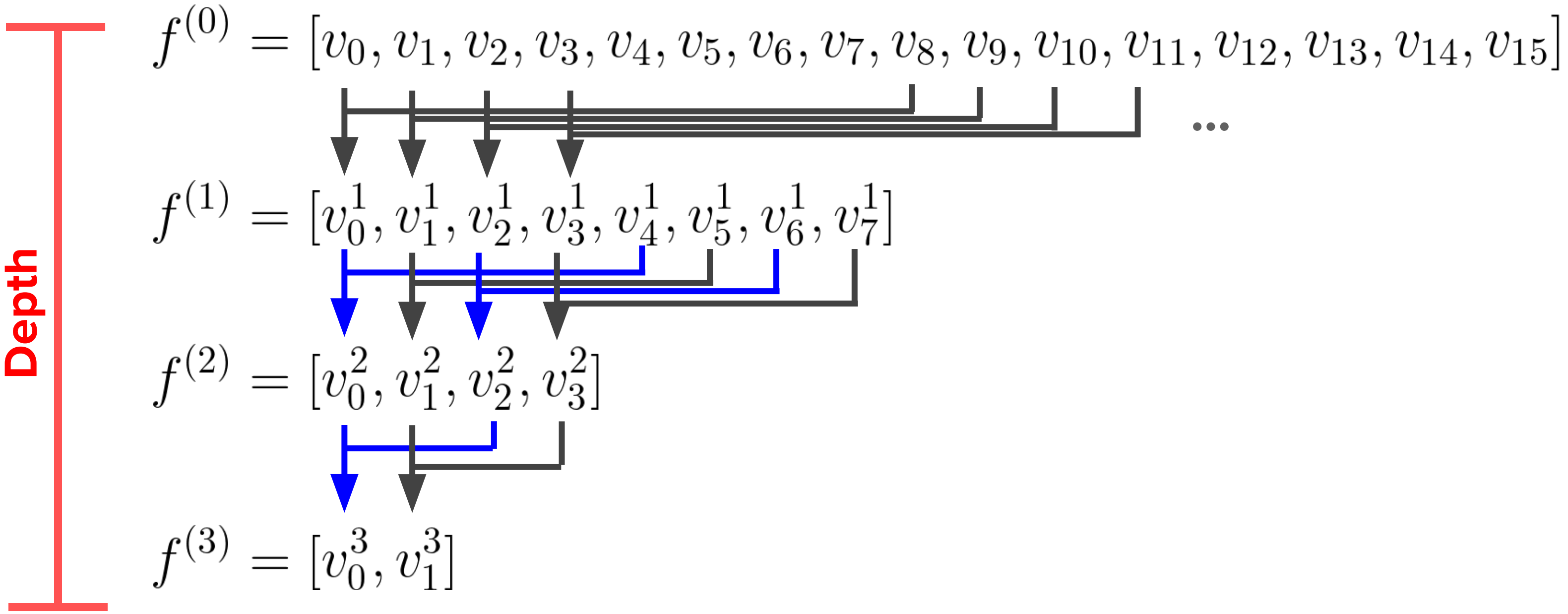
Images used in this presentation

- User faces: “Plump Interface Duotone Icons” by Streamline, Creative Commons Attribution 4.0 International, available at https://iconduck.com/sets/plump-interface-duotone-icons

- Neural network: Creative Commons Attribution-Share Alike 3.0 Unported, by Cburnett, available at https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

- FFT illustration: Creative Commons Attribution-ShareAlike 4.0, by Tikz - Alexandros Tsagkaropoulos, available at https://tikz.net/fft-algorithm-analysis/

FRI Folding



$$f^{(0)} = [v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}]$$

$$f^{(1)} = [v_0^1, v_1^1, v_2^1, v_3^1, v_4^1, v_5^1, v_6^1, v_7^1]$$

$$f^{(2)} = [v_0^2, v_1^2, v_2^2, v_3^2]$$

$$f^{(3)} = [v_0^3, v_1^3]$$

Reed-Solomon encoding

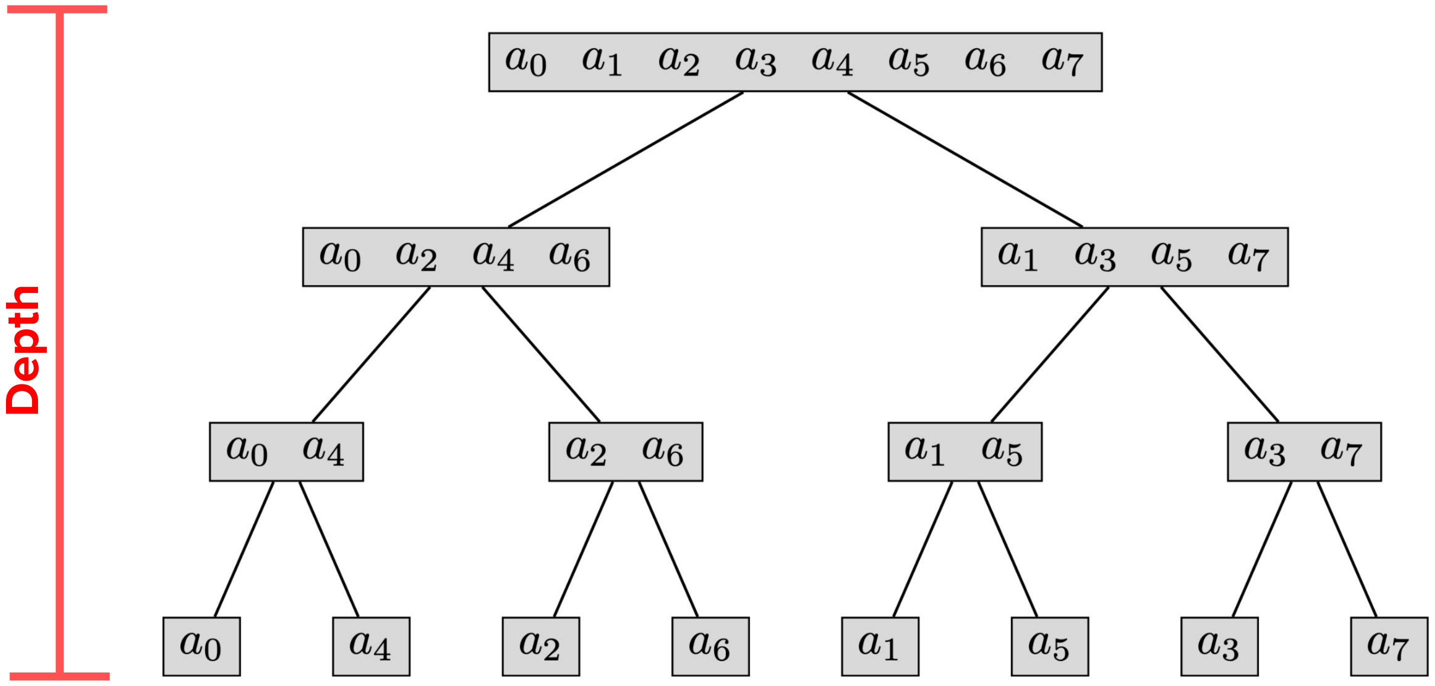
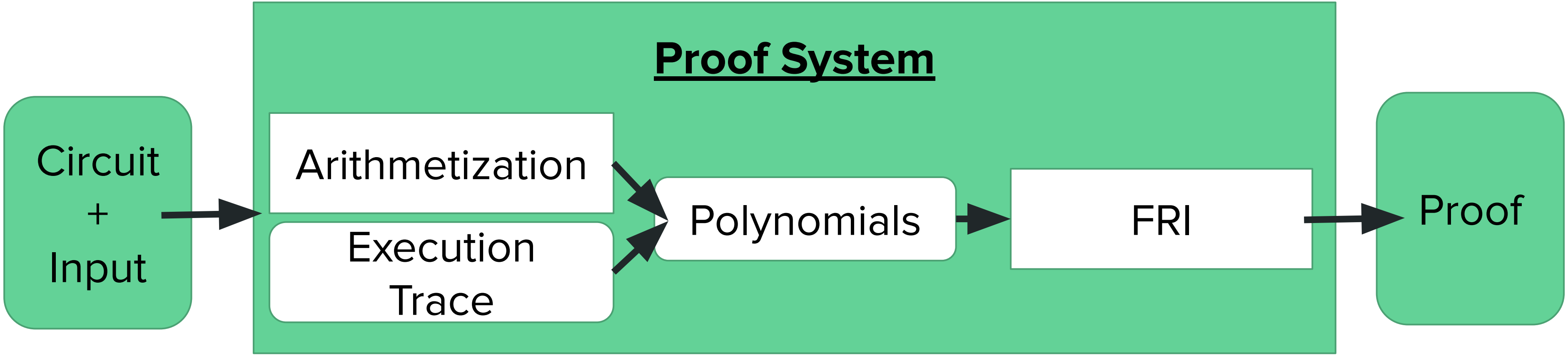


Image from: https://tikz.net/fft-algorithm-analysis/, by Tikz - Alexandros Tsagkaropoulos,

FRI


Proof System
Circuit
+
Input
Arithmetization
Execution
Trace
Polynomials
FRI
Proof