FHE Benchmarking and Beyond

NIST WPEC 2026

October 26th 2026



Shruthi Gorantala
gshruthi@google.com



Andreea Alexandru
aalexandru@dualitytech.com

Agenda



Introduction



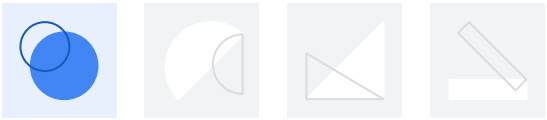
Design



Workloads



Trends and Open Calls

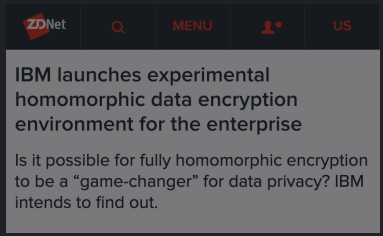


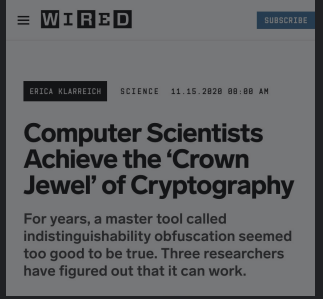
FHE Benchmarking

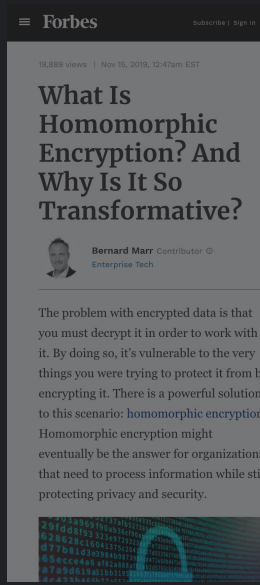
Introduction



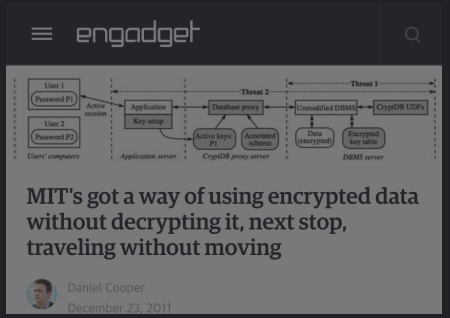




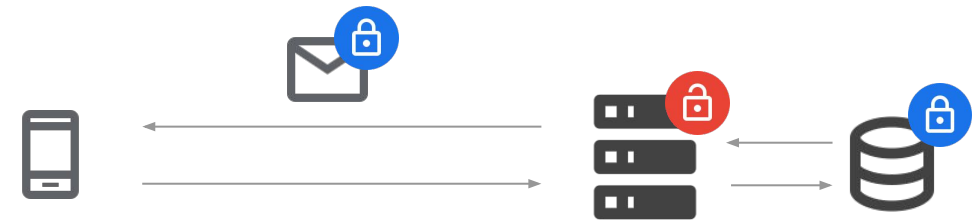




Fully Homomorphic Encryption (FHE) makes computations on encrypted data possible







Without FHE



With FHE

What problems does FHE solve?



Insider risk



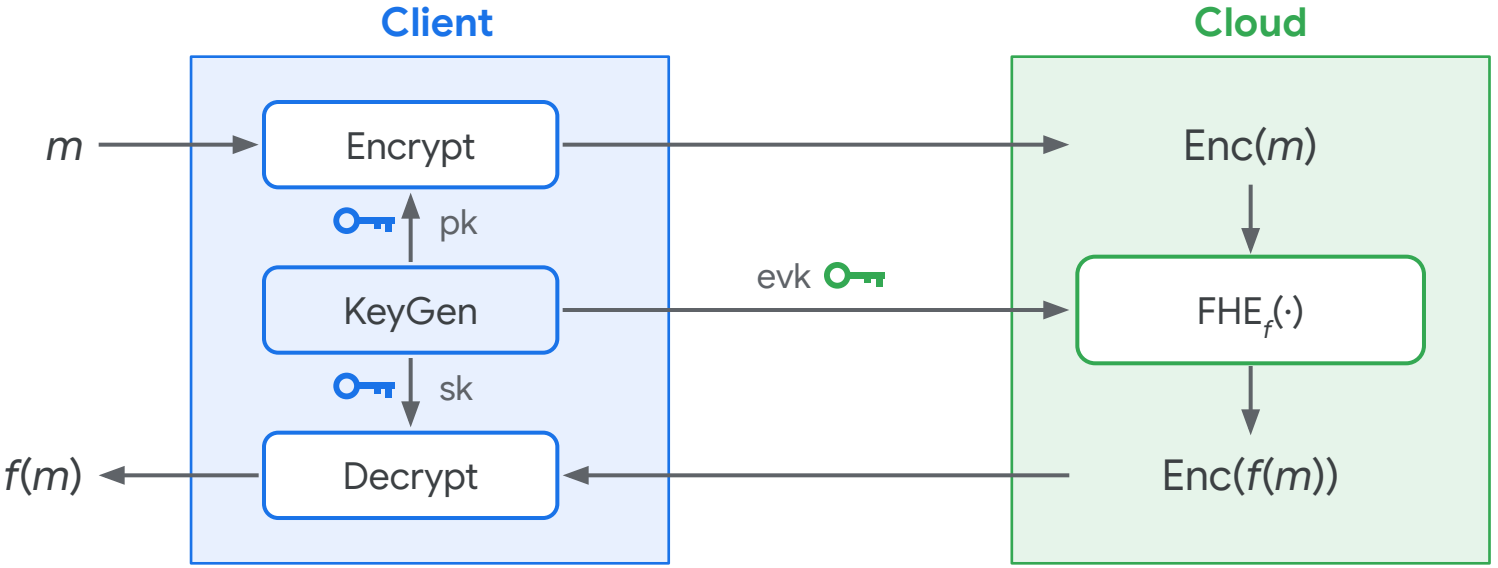
Policy compliance



Data minimization

Fully Homomorphic Encryption

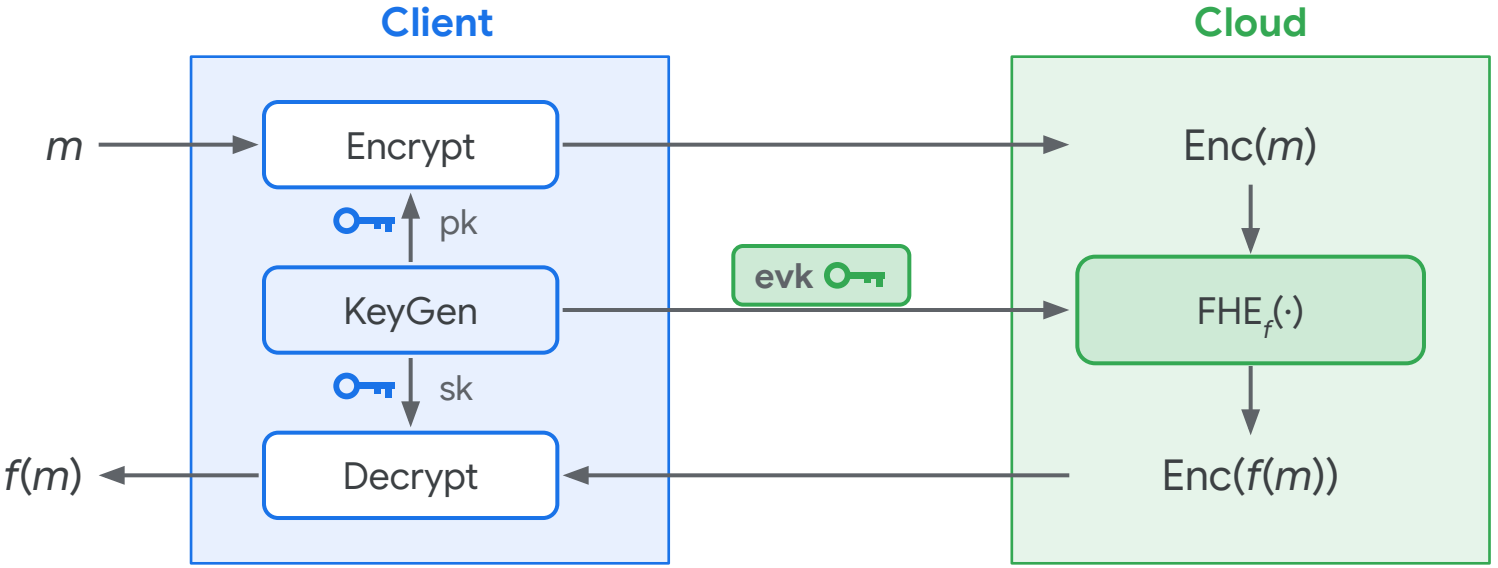
Enables **computation** on encrypted data



Delegate the **processing** of data without giving away **access** to it

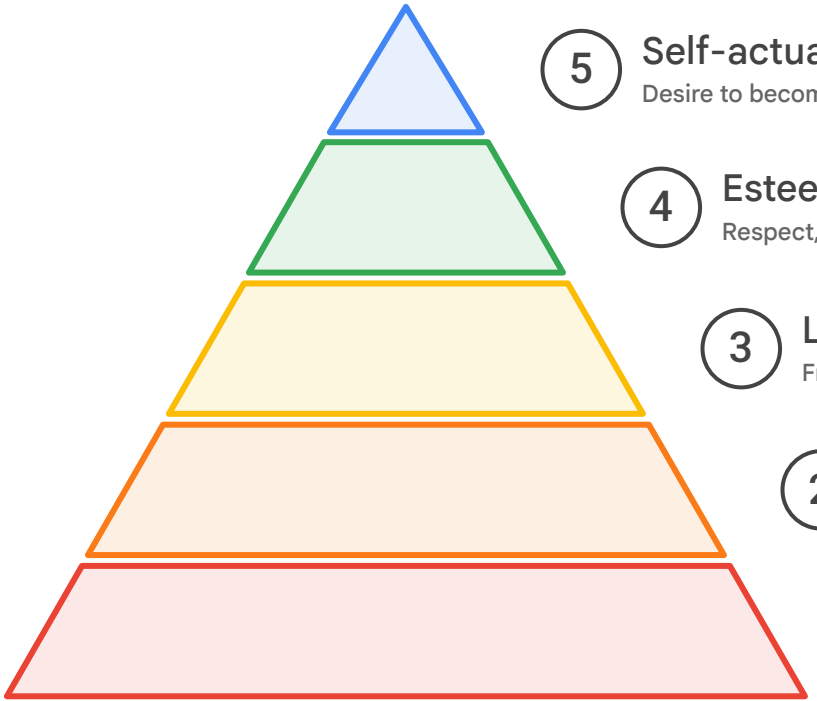
Fully Homomorphic Encryption

Enables **computation** on encrypted data



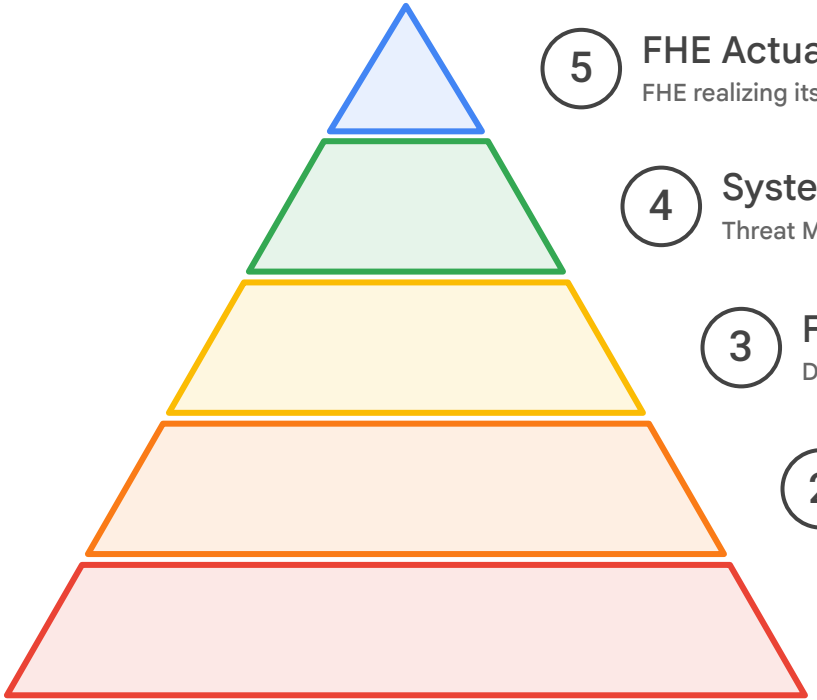
Delegate the **processing** of data without giving away **access** to it

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

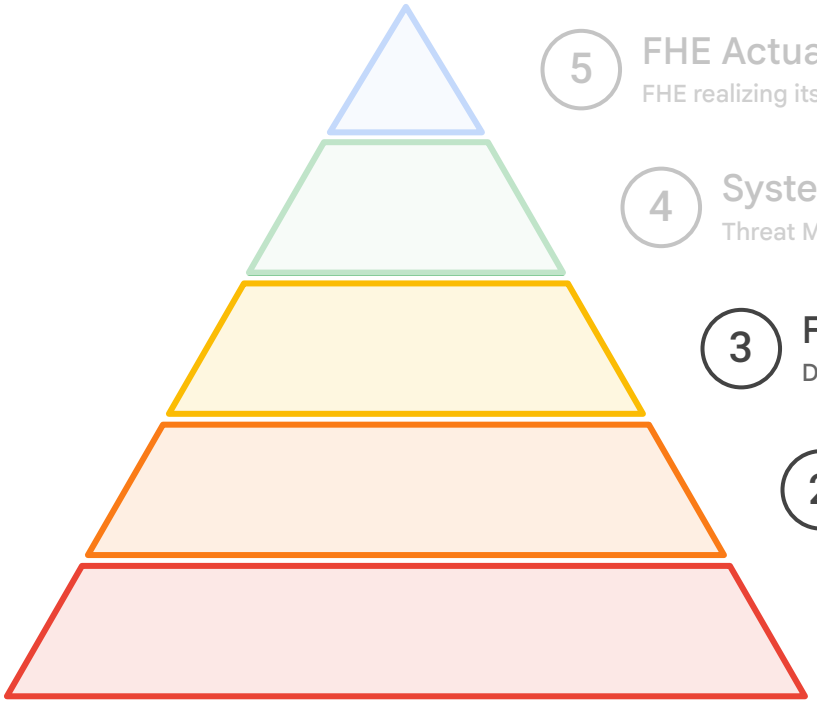
4 **Systems integration / Privacy engineering**
Threat Modeling, Privacy leaks, Network bandwidth, Observability

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 : Development



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

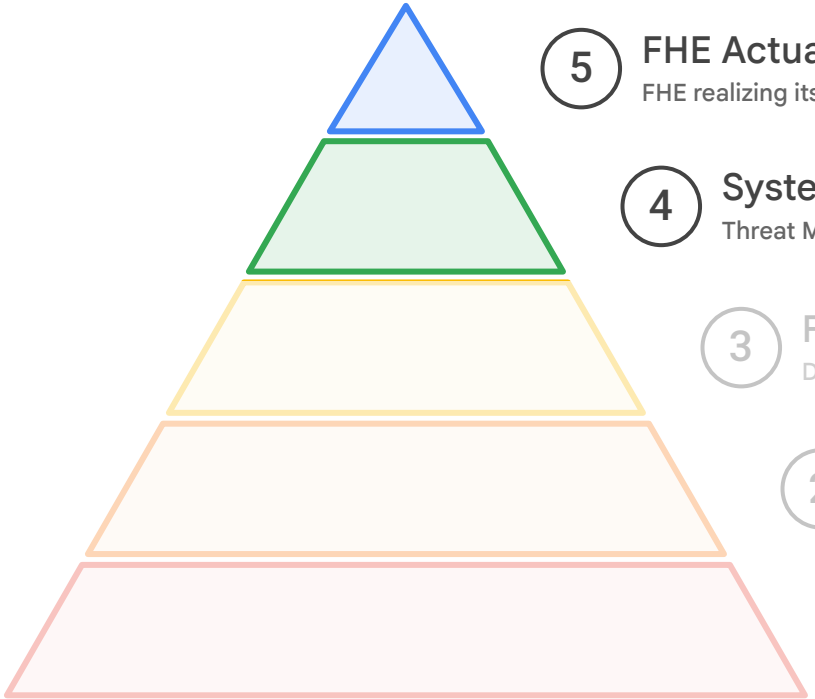
2 **Compilers / Transpilers**
Param Selection, Optimizers, Schedulers, Intermediate Representations

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

FHE Hierarchy of Needs: Development



FHE Hierarchy of Needs : Deployment



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

Need for FHE Benchmarking

- FHE is reaching practicality
  - Many applications can be feasibly developed
- FHE Deployment requires multiple performance considerations on platforms
  - Not always feasible to measure in real time
- Benchmarks on standardized platforms allow for **comprehensive and conclusive** analysis of deployment cost

Goal: Standard Benchmarking Suite

- From a technology consumer point of view
- Helps product managers/developers make decisions
  - Which of my products could feasibly use FHE?
  - Which implementation(s) of what variant(s) can I use?
- Provide useful input for FHE deployment decisions
  - End-to-end metrics
- Well-specified workloads driven by real world use cases
- Baseline reference benchmarks

Challenge: Benchmark across Software & Hardware

- Standard platforms on cloud Hardware
  - **CPU, GPU, TPU**
- Specialized Hardware
  - FPGAs, ASICs, Photonics
- **Open source and proprietary FHE** libraries and compilers
- FHE libraries in all languages (**C++, Python, Rust, Go**, etc)
- Submission from **remote services** hosted on cloud

Who can use the benchmarks?



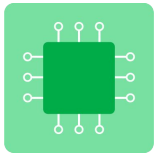
Application Developers

Interested in Application level performance for FHE Deployment



FHE Developers

Interested in FHE library and compiler based performance



Hardware Developers

Interested in Hardware Performance

What to measure?



Time



Space



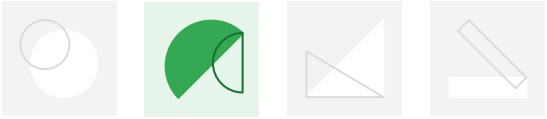
Network bandwidth



Precision



Cost



Design

Developed with HomomorphicEncryption.org



fhe-benchmarking.org

fhe-benchmarking.github.io

FHE Benchmarking Suite

Measuring Performance of Computing on Encrypted Data

About the FHE Benchmarking Suite

This Benchmarking Suite provides standardized workloads and metrics to he
feasible for their use cases. Our goal is to enable informed decisions about
comprehensive performance data across different workloads, implementatic

Each benchmark provides comprehensive metrics including:

- **Latency and Throughput** - Wall-clock time for single workload instanc
- **Memory Consumption** - Maximum RAM required during execution
- **Storage Requirements** - Keys, ciphertexts, and intermediate values
- **Communication Complexity** - Data exchanged between client and ser
- **Quality Metrics** - Accuracy loss compared to plaintext (where applicabl

Current Benchmark Workloads

Each workload represents a real-world use case with clear specifications, reference implementations, and correctness requirements.

- **Fetch-by-Similarity**. Private database queries using cosine similarity search over encrypted data.
- **ML Inference - Image Classification**. Privacy-preserving machine learning inference on encrypted images. We currently support the following datasets with more to come.
  - MNIST digit classification
  - CIFAR-10 image classification
- **Zn Multiplication**. Multiplication of two encrypted 64-bit encrypted integers. Other bit-sizes for the multiplicands will be added.
- **BERT**. Privacy-preserving BERT inference on encrypted inputs. Currently supports BERT-Base (110M) inference on the MRPC task in the GLUE benchmark.
- **Transciphering**. Homomorphic AES decryption, converting AES-encrypted data into FHE ciphertexts of the same plaintexts. The quality of the resulting ciphertexts is assessed by evaluating a mini-workload (maximum or inner product) over them.
- **Face Recognition**. End-to-end privacy-preserving face verification with encrypted feature extraction and matching, including single-pair and batched benchmark results.

Additional workloads are under development. Check back for updates.

The suite in numbers

- Officially launched in March 2026
- 6 published workloads
  - 34 benchmark variants (task x instance size)
  - 22 of them already have submissions!
- 49 published result entries in the tables: 18 from the 6 reference implementations + 31 from external submitters
- 10+ external submission packages, of which 3 are improved resubmissi

FHE Benchmarking Results - (Fetch - Small)

Column visibility

Search:

Submitter				Bandwidth			Timing (harness)					Timing (server)	
Name	Date	Env	R/L	Keys	DB	Query	Total	Keygen	DB Enc	Q Enc	Compute	Total	Compute
CryptoLab Inc.	2026-06-26 00:30:11	GPU	L	3G	128M	64K	7.7625s	3.5881s	494.8ms	21.23ms	42.1ms	19.42ms	14.84ms
Lattica-ai	2026-05-25 11:20:43	GPU	R	672M	2.3G	9M	2.7004m	12.53s	53.7s	2.074s	2.2116s		284ms
Lattica-ai	2026-02-23 21:49:44	GPU	R	1.1G	2.8G	12M	3.3609m	20.8s	54.44s	2.2528s	2.7835s		817ms
Reference	2026-02-13 22:21:16	CPU	L	2.4G	5.6G	24M	1.8759m	4.2228s	33.73s	362.6ms	1.223m	1.2167m	1.1833m

FHE Benchmarking Results - BERT (Single)

Search:

Column visibility

Timing (server) Bandwidth Timing (harness) Timing (server)

Submitter Keys Input Result Total Keygen Model Prep Input Gen Input Enc Compute Decrypt Total I/O Compute

DESILO 2026-07-14 02:04:17 GPU L 21.4G 46M 30M 14.16m 57.78s 7.0413m 13.9s 12.3s 5.3337m 1.1814s 4.9873m 2.9696m 2.0177m

Reference 2026-07-08 16:29:03 CPU L 21.4G 46M 30M 41.83m 51.62s 3.2822s 4.8858s 3.9553s 40.62m 633.3ms 40.24m 58.37s 39.27m

FHE Benchmarking Results - Zn Multiplication (Small Batch - 1000)

Column visibility

Search:

Name Submitter Bandwidth Timing (harness) Timing (server)

Date Env R/L Keys Input Result Total Keygen Input Gen Input Enc Compute Decrypt Total Compute

CryptoLab Inc. 2026-08-13 20:27:39 CPU L 928.5K 1.7M 400.1K 211.3ms 15.3ms 117.8ms 14.8ms 47.03ms 15ms 25.47ms 9.08ms

CryptoLab Inc. 2026-08-12 05:27:29 CPU L 1.5M 4M 2M 222.6ms 15.2ms 107.6ms 26.8ms 54.17ms 17.87ms 38.43ms 13.03ms

Hongren Zheng 2026-03-02 05:48:29 CPU L 2.3M 8M 2M 668.1ms 71ms 121.3ms 144.6ms 138.4ms 105.3ms

Reference 2026-02-20 03:37:04 CPU L 125M 1004.7M 502.3M 17.95m 800.8ms 129.4ms 7.4931s 17.8m 375.4ms

Suite Structure

HARNESS

- Drives every stage
- Generates datasets and inputs
- Collects measurements, including io artifacts (keys, ciphertexts, etc.)
- Specifies validation, verifies and scores results

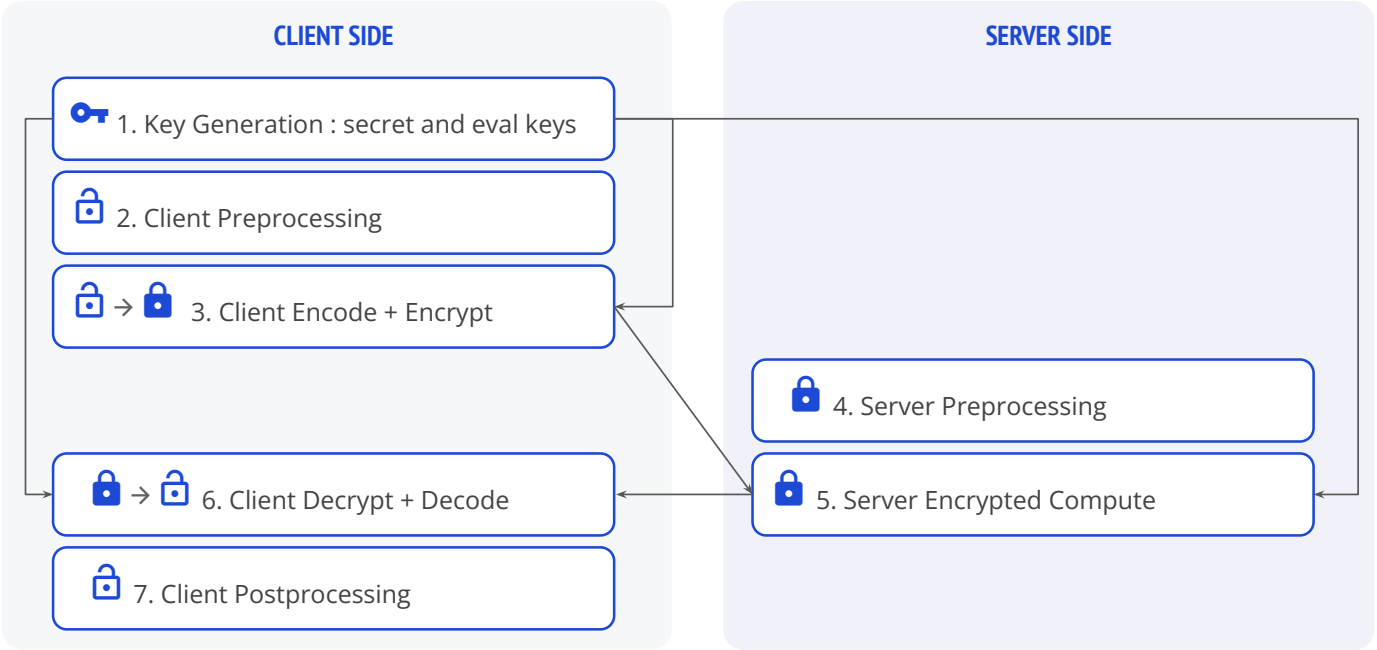
ONE REPOSITORY PER WORKLOAD

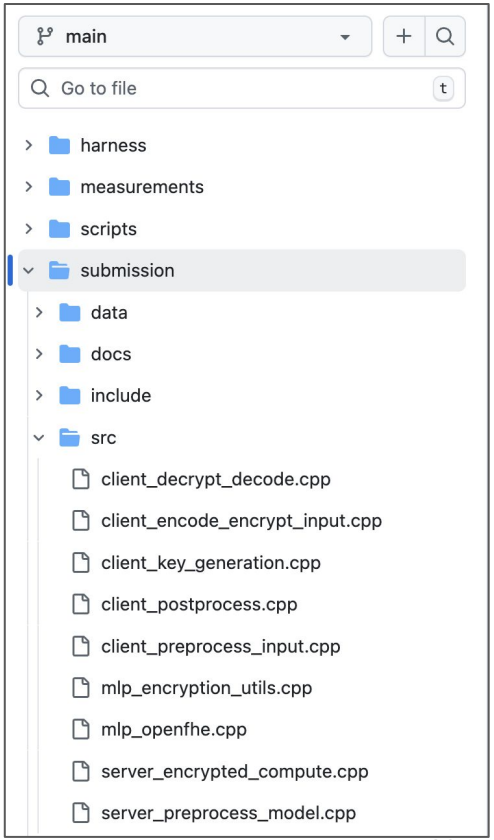
- **datasets/** *generated inputs*
- **harness/** *drives + measures*
- **io/** *keys, ciphertexts, results*
- **measurements/** *timings + sizes*
- **scripts/** *build + setup*
- **submission/** *code stages*

SUBMISSION

- Open or closed source
  Local or remote
- Any language
- CPU, GPU, accelerators
- Have to provide executables and generate artifacts where specified

Suite structure: Two-party non-interactive protocol





Suite Structure : Harness

PREPARE

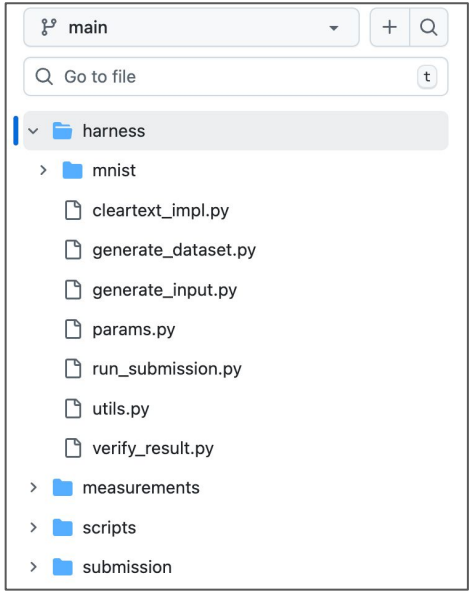
- Generate data + model
- Generate input
- Cleartext reference

EXECUTE

- Run client + server exes
- Measure every stage
- Run multiple iterations

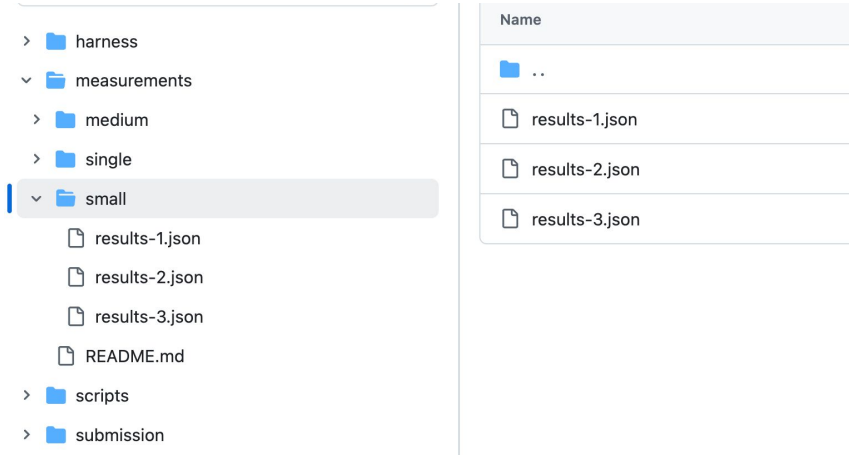
EVALUATE

- Verify result
- Score quality
- Save results



Suite Structure : Measurements

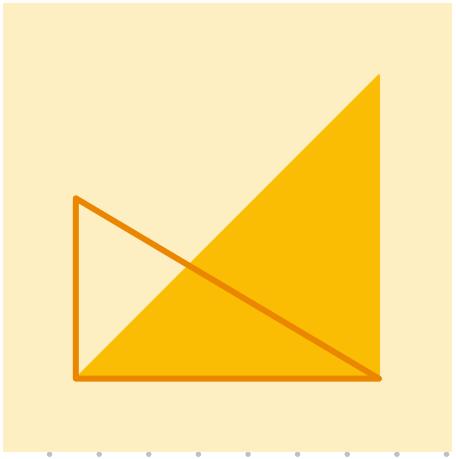

```
{
  "Timing": {
    "Test dataset generation": "4.7884s",
    "Key Generation": "0.9751s",
    "Encrypted model preprocessing": "0.0037s",
    "Input generation": "1.8093s",
    "Input preprocessing": "0.0035s",
    "Input encryption": "0.0936s",
    "Encrypted computation": "7.6682s",
    "Result decryption": "0.0842s",
    "Result postprocessing": "0.0028s",
    "Total": "15.4287s"
  },
  "Bandwidth": {
    "Public and evaluation keys": "1.0G",
    "Encrypted input": "5.0M",
    "Encrypted results": "1.0M"
  },
  "Server Reported": {
    "Encrypted computation": "6.515s",
    "Total": "6.515s"
  }
}
```



Workloads

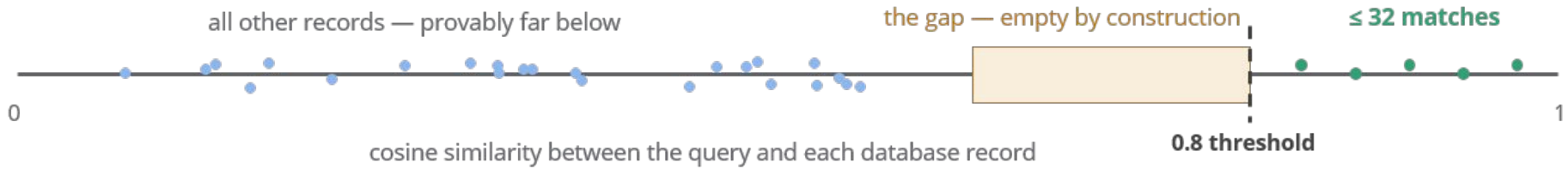
Tales from the field



Fetch by Similarity

Private k-nearest-neighbor vector-search over encrypted inputs, using cosine similarity
Essential for private biometrics and private RAG

- Both db and query are *encrypted* → *potentially large storage required*
- Deliberate simplification: *promise of gap in score and at most 32 matches above threshold*
  - The gap makes encrypted comparison cheap
  - < 32 matches make the response fixed-size



Fetch by Similarity

Workload variants

Count – how many records match per query

Fetch – return payloads of the matches (64-bit payloads)

#recs x dim – **Small**: 50K x 128, **Medium**: 1M x 256, **Large**: 20M x 512 (*no runs yet*)

Search on 1M records: CPU reference vs fastest GPU submission

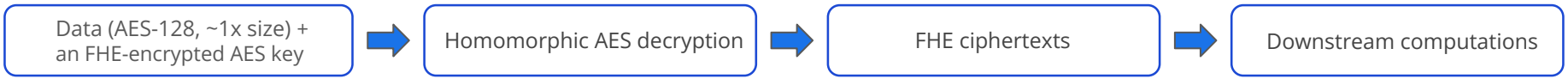
- Count: 65x reduction in end-to-end runtime (28.9 s) and >40x reduction in enc db size and enc query (4 GB, 64 KB)
- Fetch: retrieval costs 1.4x–1.7x more than count in server compute

Once the server is fast, the client dominates – one-time medium DB encryption is half of the end-to-end run: fixed-cost vs per-query cost

AES-FHE Transciphering

Bridge symmetric encryption and FHE: upload data AES-encrypted and compute under FHE

Essential for bandwidth-constrained clients and for data already encrypted at rest



- The quality of the transciphered ciphertexts (noise, remaining levels) is not straightforward

  ⟶ *Mini-workloads as downstream applications to test end-to-end performance/accuracy*

Maximum – comparison-heavy

Inner product mod 2^{16} – arithmetic-heavy

Toy: 1 block – ECB, **Small**: 16 – CTR,

Medium: 256 – CTR, **Large** 4096 – CTR

The number of blocks amortizes over the conversion + The conversion amortizes over the workload

Large-Integer Multiplication

Exact multiplication of encrypted 64-bit integers (mod 2^{64})

Essential for blockchain-style applications: many exact arithmetic operations

- Workload variants

  Single – one multiplication ⟶ latency

  Batched (Small, Medium, Large) – 1 K / 100K / 10M pairs ⟶ throughput

- The all-CPU leaderboard
  - Single: latency 2.34 s ⟶ 181 ms, key material 125 MB ⟶ 54 KB
  - 10M: latency 28.1 min ⟶ 52.7 s, encoding 76.6 GB ⟶ 16.4 GB, results 19.2 GB ⟶ 3.7 GB

*Batching in action: per-multiplication cost falls **~34,000×** from a single call to a 10 M batch.*

ML Inference workloads



MNIST

Digit Detection

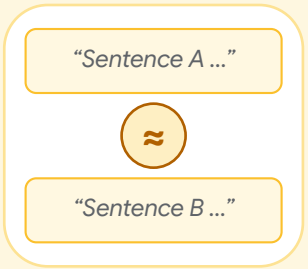
MLP



CIFAR-10

Object Classification

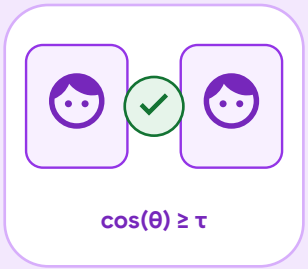
ResNet-20



BERT

Sentence Equivalence

BERT-Base Transformer

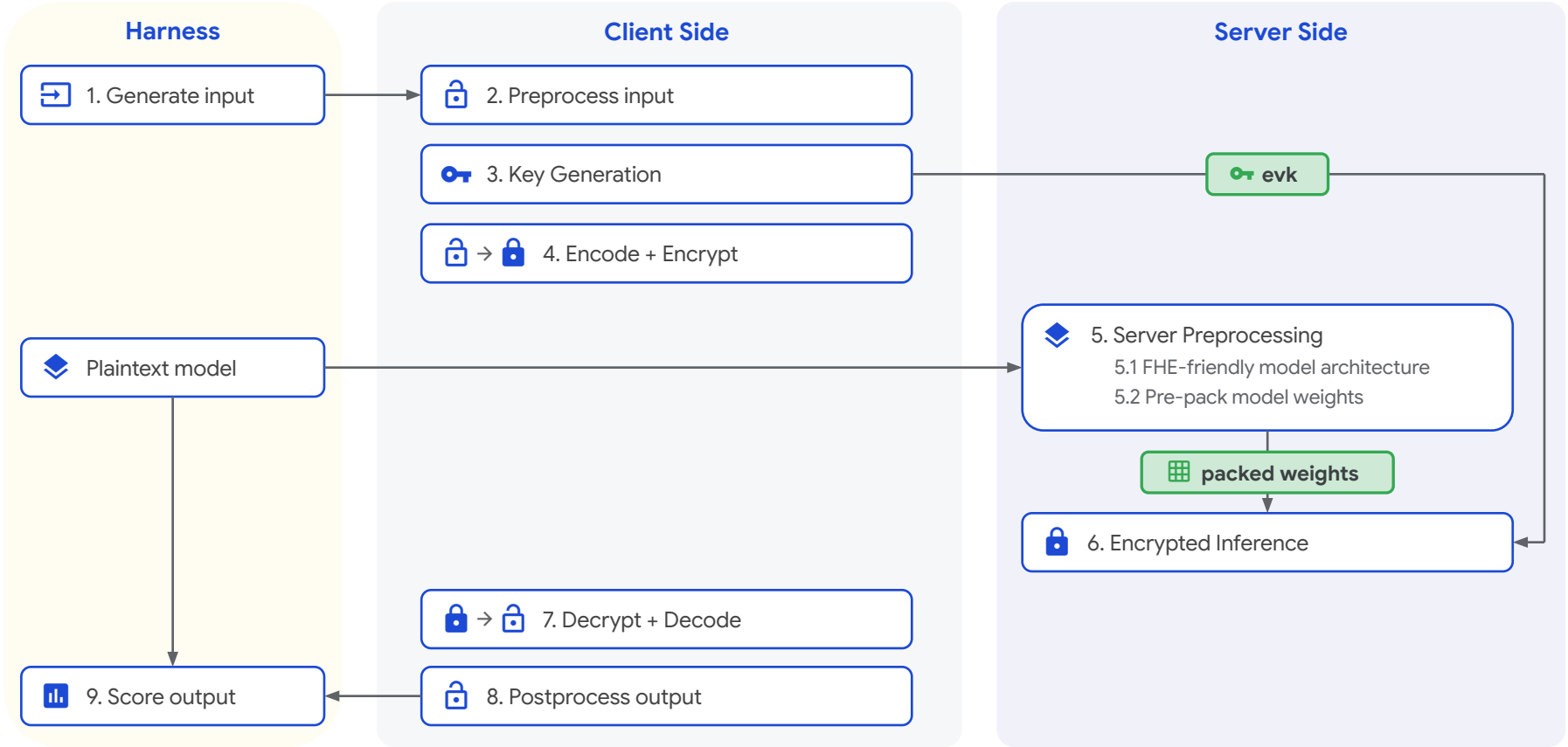


Face Recognition

1:1 Face Verification

CryptoFace

ML Inference: Two-party non-interactive protocol



ML Inference

Fixed by the Workload

- **Dataset**
- **Quality threshold**
- **Single-query inference** to measure latency
- **Batched inference** to measure throughput

Free for the Submitters

- **Custom model architectures**
- **Client pre- / post-processing**
- **Cryptographic trade-offs:** evaluation key sizes, ciphertexts, and compute

Supported Submission Models

Open Source

Full source in submission/

(e.g., **Reference baselines**)

Closed Source

Pre-compiled library/binary

(e.g., **DESILO, Cryptolab**)

Remote Backend

Client shims to cloud/HW backend

(e.g., **Lattica**)

ML Inference - MNIST

Workload variants

Single – one 28×28 digit image (0–9)

Batched – quality bar: ≥90% accuracy

#images (N) – **Small**: 100, **Medium**: 1K, **Large**: 10K (*all 4 variants run*)

MNIST (MLP): CPU reference vs. fastest GPU submission

- **Single Inference:** 6 secs on CPU to 0.567 ms on GPU
- **1K batch: 229× speedup** (1.83 h ⟶ 28.8 s total, 0.568 ms compute)
- **10K batch (GPU): 2 ms total**

Encrypted accuracy (97.2%) matches plaintext (97.4%) at ~0.2–0.6 ms GPU compute/batch

CIFAR-10

Workload variants

Single – classify one 32×32 RGB image into 10 object classes (CIFAR-10)

Batched – classify a batch of RGB images (quality bar: ≥90% accuracy)

#images (N) – **Small**: 100, **Medium**: 1K, **Large**: 10K (*batches: no runs yet*)

CIFAR-10 object classification (ResNet-20): CPU reference baseline by **FHEON**

- Single inference (CPU): **32 min** encrypted compute on CPU

Deep residual networks (ResNet-20) grow evaluation keys to ~50.4 GB

BERT Inference

Workload variants

Single – classify 1 sentence pair for semantic equivalence (MRPC)

Batched – classify a batch of pairs (quality bar: ≥75% accuracy)

#pairs (N) – **Small**: 100, **Medium**: 200, **Large**: 408 (*Med/Lrg: no runs yet*)

BERT-Base (110M) on MRPC: CPU reference vs GPU submission (**THOR - DESILO**)

- Single inference: 20x reduction in server compute (39.3 min ⟶ 2 min) with GPU
- Small batch (100 pairs, GPU): 3.32 h compute (6.37 h server total) at 84.3% accuracy (vs ≥75% bar)

*On GPU, encrypted BERT inference is I/O-bound: 3.0 of the 5.0 server minutes are spent moving **21.4 GB of keys** and **110 GB of packed weights***

Face Recognition - Use cases

Biometric Authentication: Verify encrypted facial templates without exposing raw biometrics.

Face Unlock

Phone & app biometric login on user devices

Output → Client

Account Recovery

Step-up biometric check to restore locked accounts

Output → Server

Retail Autopay

Checkout-free store entry & pay

Output → Server

ID Verification

Identity & age/travel verification

Output → Server

Server-Side Output Reveal: Unlike standard client-side inference, most authentication flows require **revealing the match decision at the server** to authorize access or payment.

Face Recognition

Workload variants

Single – verify 1 **CelebA** face pair via encrypted similarity score

Batched – verify a batch of pairs

#pairs (N) – **Small**: 128, **Medium**: 256, **Large**: 1,024 (*all 4 variants run*)

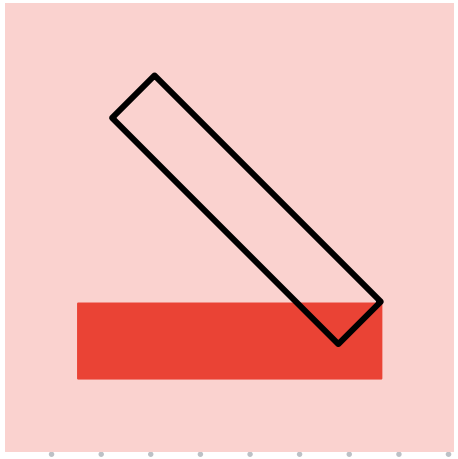
CelebA face verification: **CryptoFace**

- Single inference: 2 mins (on CPU) to 15 secs (GPU)
  - **114.1 GB packed encrypted model + 28.8 GB keys**
- Batched: 49.5 min (Small) to 6.07 h (Large); TAR@FAR=1% is 0.84–0.85 vs 0.94 ArcFace

*The encrypted model weights are: **113 GB.** GPU implementation on single pair takes **15 seconds***

Trends and Open Calls

What’s coming, what we still need, who can contribute



What's Next: Emerging FHE Directions

Emerging directions shaping the next wave of **FHE workloads and harnesses**:



GenAI

Private LLM Inference

Encrypted LLM Inference of open sourced LLM models.



Retrieval

Private RAG & PIR

Private RAG on a public database using Private Information Retrieval.



Analytics

Private Join & Compute

Intersecting private datasets & computing encrypted aggregates across parties.



Security

Private Malware Discovery

Scanning telemetry or binaries against encrypted threat signatures.



Training

Private Training

Encrypted fine-tuning & collaborative ML across institutions.



AI for FHE

AI-Based Optimizations

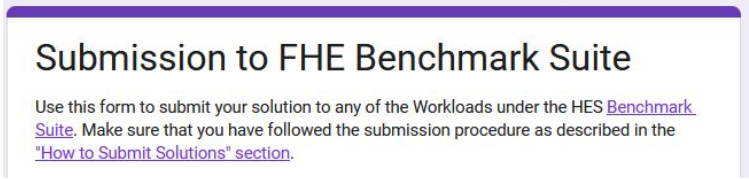
Standardized benchmarks as reward signals for AI circuit & kernel optimization.

New Workloads

- LLM Inference
- PIR
- FHE based PSI
- Private ML Training
- Encrypted regex
- Database queries
- ...

How to submit

- **Fork Repository github.com/fhe-benchmarking/<workload>.**
- **Implement Workload**: populate /submission subdirectories
- **Document Solution**: update the READMEs with all relevant information and optionally provide more documentation in the /docs subdirectory.
  - Justification of achieving 128-bit security level, < 2^{-40} probability failure
- **Generate Measurement Files**: run the harness with a --num_runs 3 argument
- **Submit Form**: make the fork public and notify suite organizers via the **form** on the website; *sign author certification*



The future of submissions?

The harness is a ready-made agent loop: generate a solution, run it, get a verdict and re-iterate
→ *an evaluation environment built for FHE engineering, useful for agents out of the box*

The fuel: state-of-the-art results land quicker in open-source libraries, free for anyone to build on

- Agent-generated, working submissions (CRYPTO PPML 2026) built around hand-written expert FHE skills → *frontier models keep advancing, needing expert skills less and less*

The catch: the harness verifies correctness and quality, not the security of the implementation

- A cheating solution can pass every check while leaking → *today's guarantee is the author's certification*
- How to automatically verify submissions?

Open problems for standardization

The suite requires submissions at 128-bit security and 2^{-40} probability failure, but questions are still open

Fixed, agreed parameters

- Currently different secret key distributions are used, failure probability hard to consistently define
- Need recommended parameters profiles the harness can automatically check

Active security

- Honest-but-curious server assumption
- Need integrity against a malicious server and to measure its cost

Cost of security: solving this issues costs performance – how much?

Call to action: harness improvements, new workloads, new submissions, maintainers

Thank you

fhe-benchmarking.org

Contributors are welcome!

fhe-benchmarking@homomorphicencryption.org
gshruthi@google.com
aalexandru@dualitytech.com