FHE-Coder: Benchmarking Secure Agentic Code Generation for Fully Homomorphic Encryption

Mayank Kumar, Jiaqi Xue, Mengxin Zheng, Qian Lou

***International Conference on Learning Representations* 2026,**

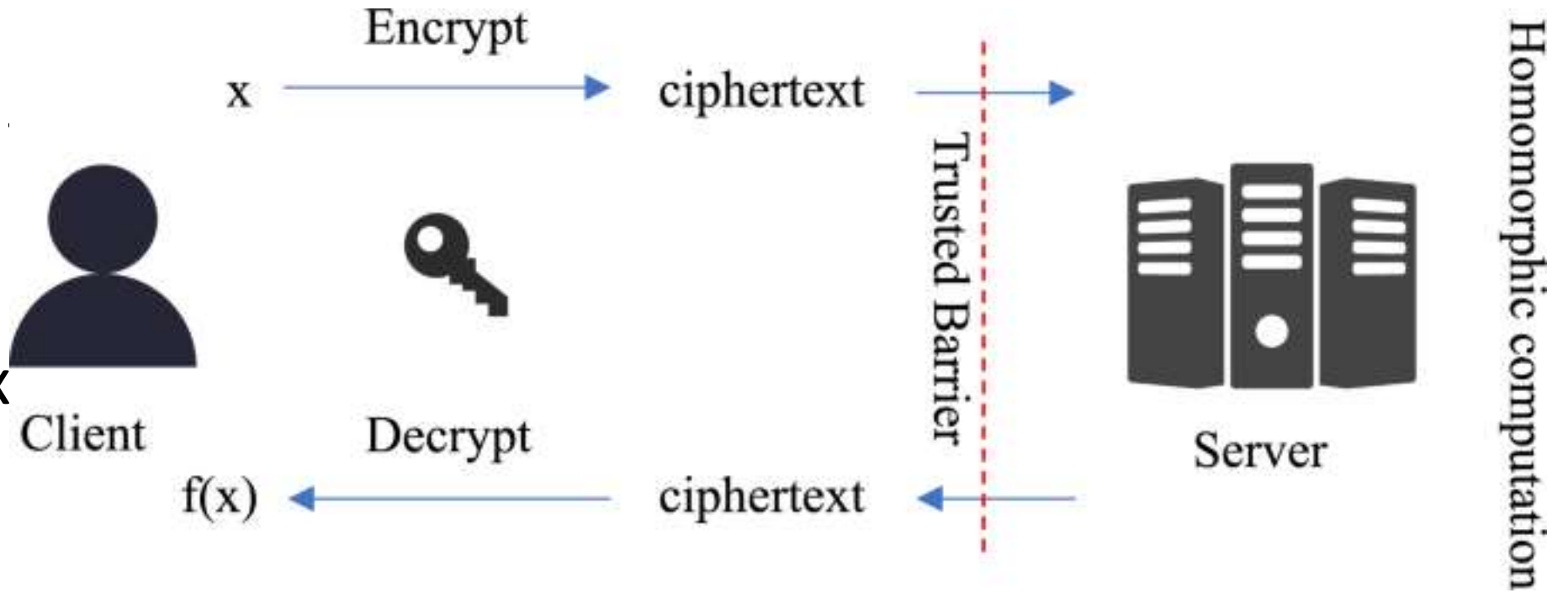
Rio De Janeiro

What is Fully Homomorphic Encryption and why should you care?

Enc(x)

Enc(x)

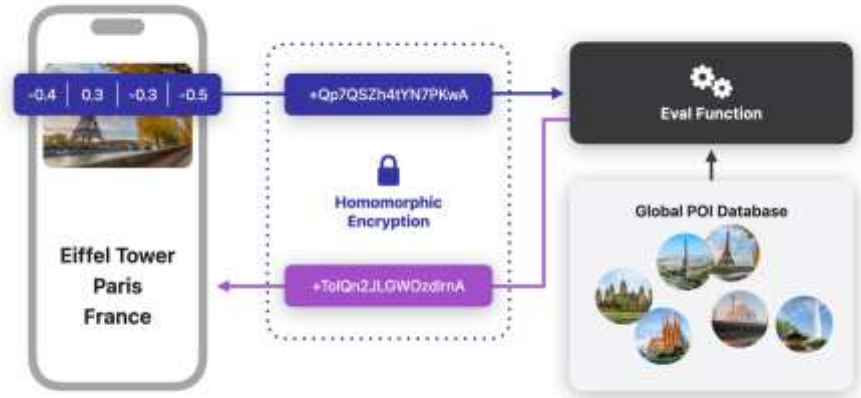
f(Enc(x



Can be an arbitrary circuit!

Image Source: https://link.springer.com/article/10.1186/s42400-025-00360-x

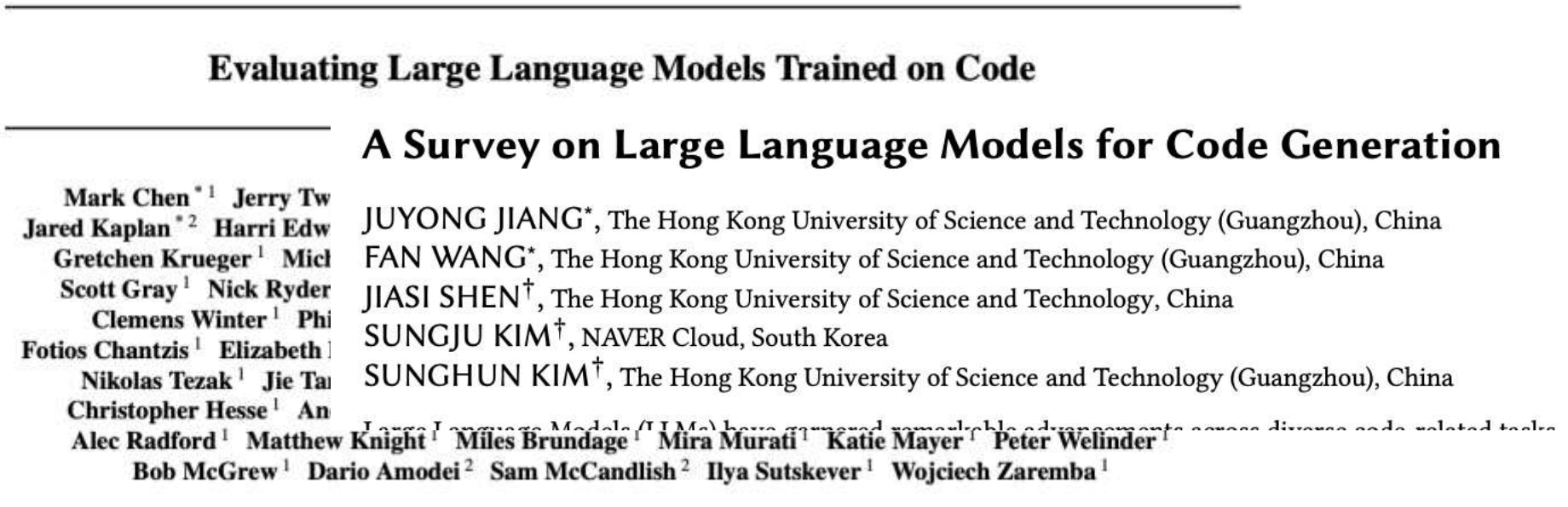
What is Fully Homomorphic Encryption and why should you care?



Apple iOS 18

Microsoft Edge Password Monitor

Language model-based code generation offers a practical solution.



Evaluating Large Language Models Trained on Code

Mark Chen [*1] Jerry Tw
Jared Kaplan [*2] Harri Edw
Gretchen Krueger [1] Mich
Scott Gray [1] Nick Ryder
Clemens Winter [1] Phi
Fotios Chantzis [1] Elizabeth
Nikolas Tezak [1] Jie Ta
Christopher Hesse [1] An
Alec Radford [1] Matthew Knight [1] Miles Brundage [1] Mira Murati [1] Katie Mayer [1] Peter Welinder [1]
Bob McGrew [1] Dario Amodei [2] Sam McCandlish [2] Ilya Sutskever [1] Wojciech Zaremba [1]

A Survey on Large Language Models for Code Generation

JUYONG JIANG*, The Hong Kong University of Science and Technology (Guangzhou), China
FAN WANG*, The Hong Kong University of Science and Technology (Guangzhou), China
JIASI SHEN†, The Hong Kong University of Science and Technology, China
SUNGJU KIM†, NAVER Cloud, South Korea
SUNGHUN KIM†, The Hong Kong University of Science and Technology (Guangzhou), China

Large Language Models (LLMs) have garnered remarkable advancements across diverse code-related tasks

Language model-based code generation offers a practical solution.

Can LLMs write secure encryption code?

❌ **Plaintext Code**

```
int result = a & b; // No encryption!
```

✓ **FHE Code**

```
bootsAND( &result, &a,&b, key->cloud);
```

Without our framework, LLMs almost always produce the left — not the right.

The Challenge: FHE Code is Uniquely Hard to Write

Parameter Hell

- LWE parameters (λ, lattice dim, modulus) must be tightly coupled
- Even a slight misconfiguration silently breaks the security guarantee

API Minefield

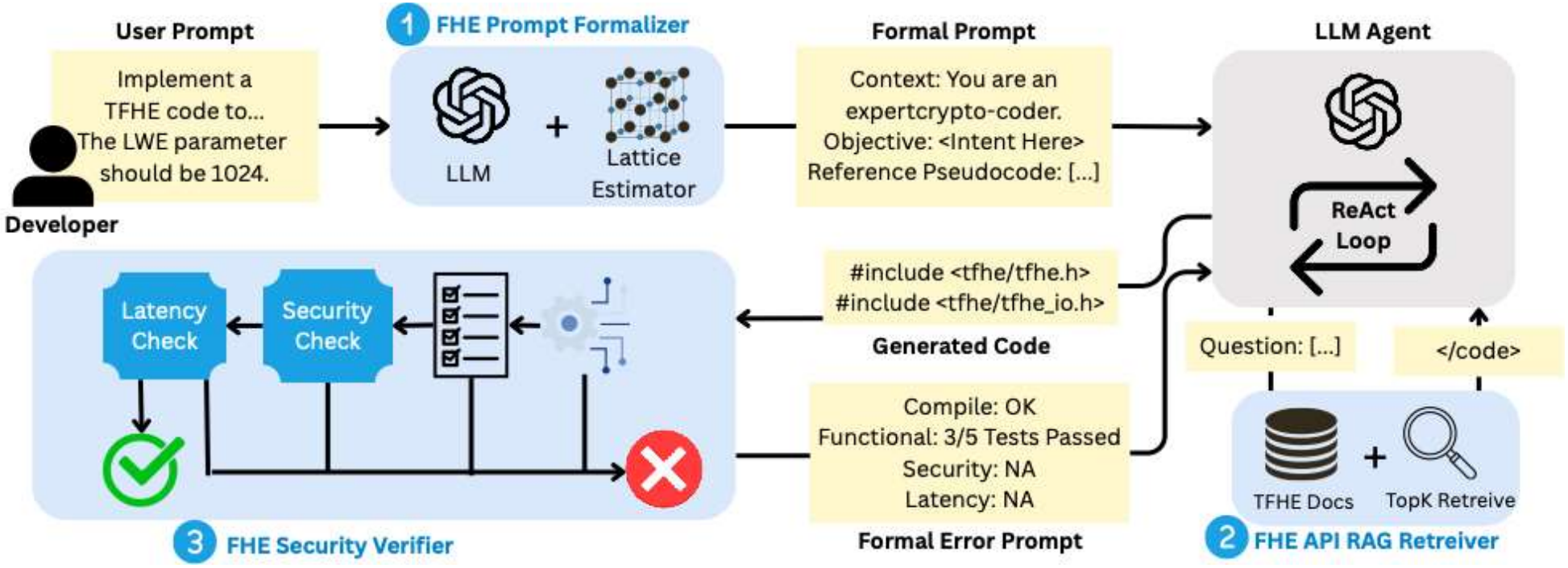
- TFHE/CKKS APIs have strict noise constraints and non-obvious argument orders
- LLMs hallucinate functions or misordering args, producing broken ciphertext ops

False Positives

- Plaintext code compiles, runs, and passes unit tests
- Functional tests have no visibility into whether encryption was used at all

Key Insight: Pass@k(func) is a misleading proxy — a program can pass all tests while being completely insecure.

FHE-Coder: A Three-Phase Agentic Framework



ReAct Agent Loop • Max 10 iterations • Works with GPT-5, Gemini-2.5-Pro, Qwen3-Coder, DeepSeek-V3.1

New Metric: Pass@1(func_sec) + 13-Task Benchmark

Why Functional Tests Aren't Enough

Pass@k(func)

Passes unit tests — may be plaintext!

Pass@1(func_sec) ★

Functional AND cryptographically secure

A code is secure only if it:
(i) uses only TFHE/CKKS APIs
(ii) passes LWE parameter bounds
(iii) encrypts all inputs before use

Benchmark Tasks (TFHE + CKKS)

Primitives

AND, ReLU, Adder, Multiplier

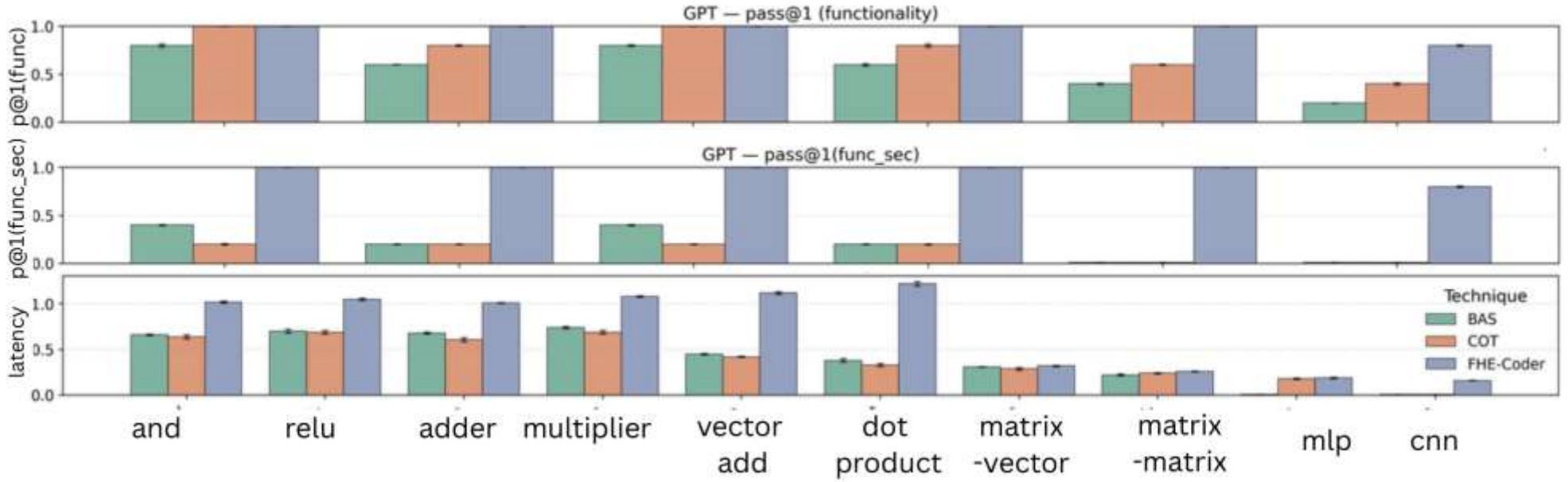
Linear Algebra

Vector Add, Dot Product, Mat-Vec, Mat-Mat

Deep Learning

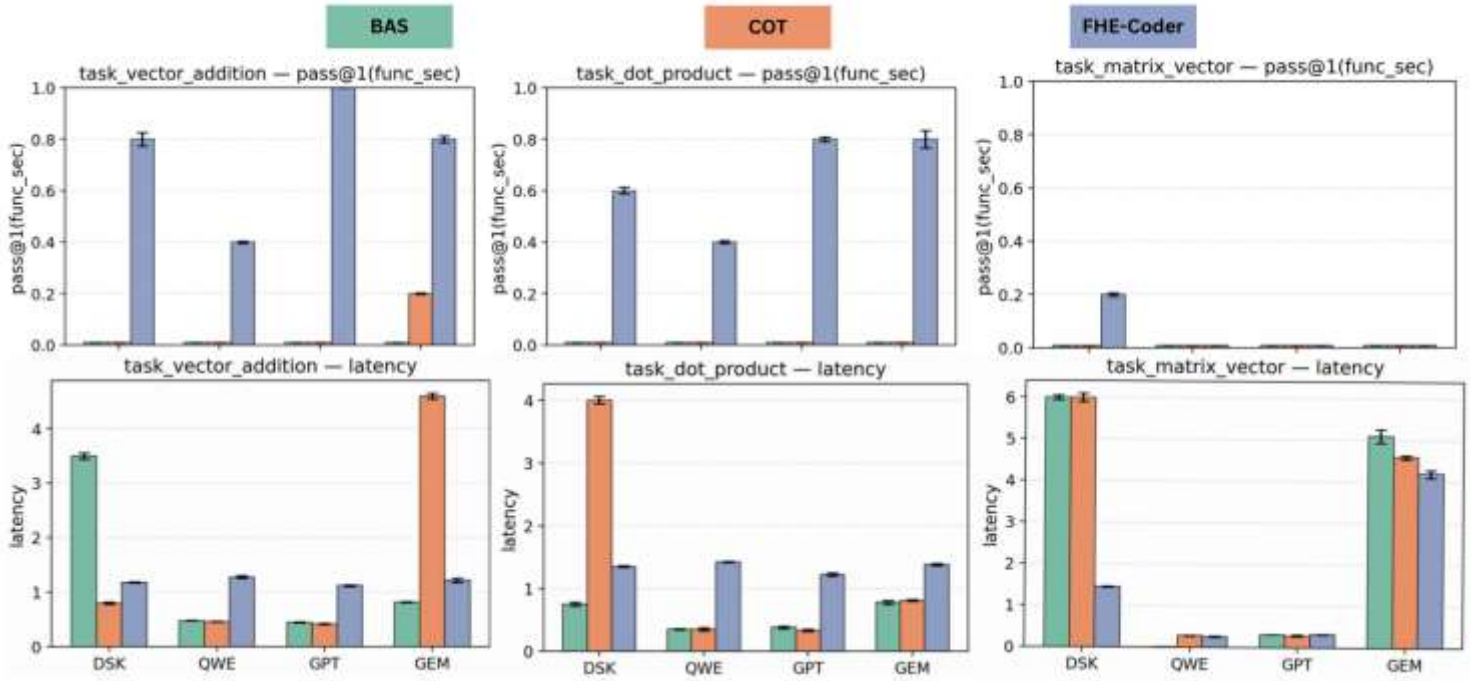
MLP, CNN, Softmax, Attention, Transformer

Baselines Pass functional tests but fail security, FHE-Coder achieves Both!



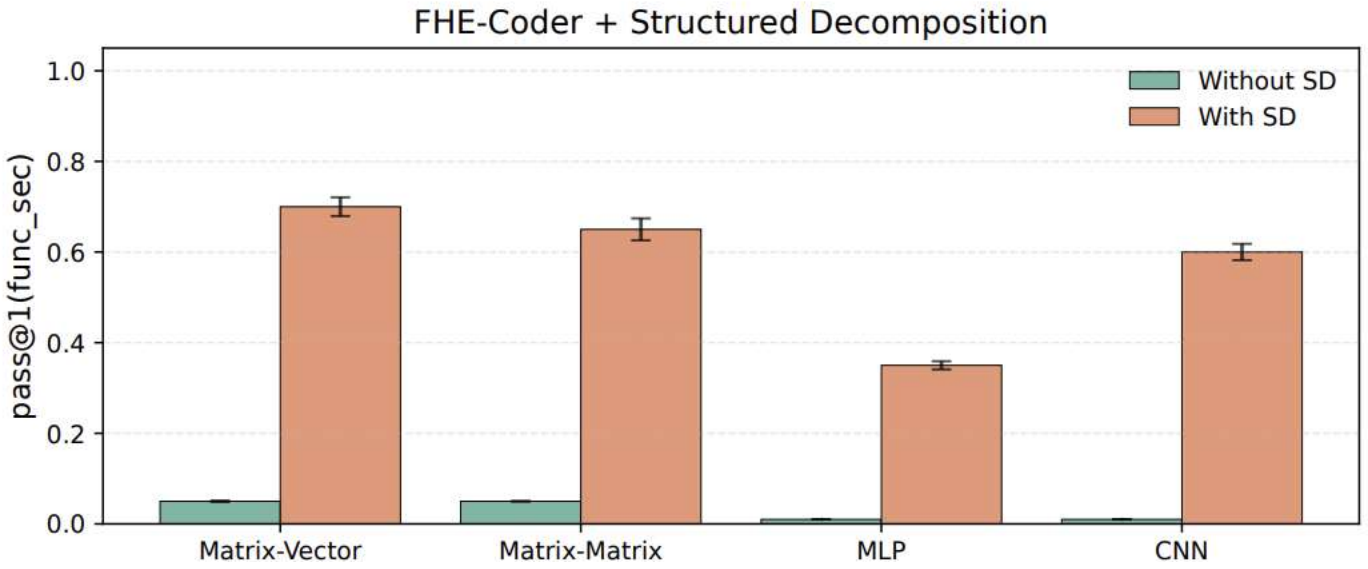
Performance of GPT-5 across representative tasks. A comparison of our framework (FHE-Coder) against Baseline (BAS) and Chain-of-Thought (COT) techniques. FHE-Coder consistently produces code that is both functionally correct and verifiably secure.

Baselines fail security often, FHE-Coder Improves Every LLM



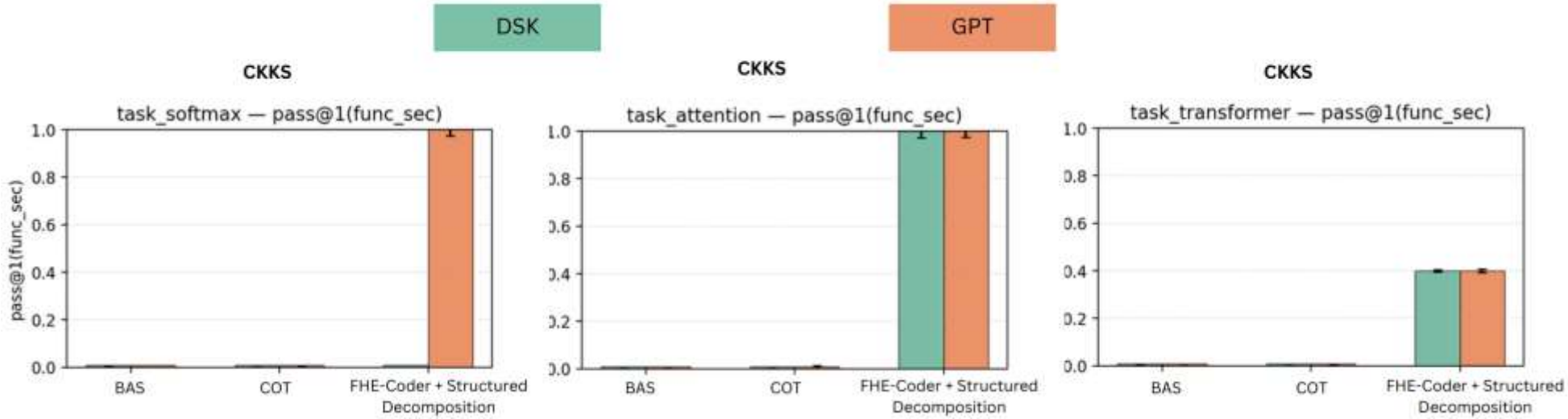
Comparison of FHE-Coder against baselines (BAS, COT) across four LLMs. Baselines universally fail security checks, whereas FHE-Coder consistently ensures security, showing model-agnostic benefits. Note: Latency measures the relative runtime overhead of the generated code.

For harder tasks, human guidance via structured decomposition is needed



Performance: FHE-Coder + structured decomposition for complex tasks

FHE Coder generalizes to CKKS scheme as well



Generalization to CKKS. FHE-Coder + Structured Decomposition significantly outperforms failing baselines on non-linear architectures. While GPT achieves high pass@1(func sec) across all tasks, DSK's failure on Softmax indicates that the framework's effectiveness remains partially constrained by the base LLM's intrinsic capabilities.

Conclusion

First FHE Benchmark

13-task benchmark spanning TFHE and CKKS schemes, covering primitives, linear algebra, and deep learning workloads. The first rigorous evaluation suite for agentic FHE code generation.

Pass@1(func_sec) Metric

Standard Pass@k(func) is a misleading proxy. Our joint metric requires code to be both functionally correct and cryptographically secure, exposing silent insecurity that tests alone cannot detect.

Framework Consistently Wins

FHE-Coder's three-phase pipeline (Formalizer + RAG + Security Verifier) succeeds where BAS and COT baselines universally fail, grounding generation in LWE mathematical hardness.

Model-Agnostic & Extensible

Validated on GPT-5, Gemini-2.5-Pro, DeepSeek-V3.1, and Qwen3-Coder. Modular design generalizes to CKKS and scales to complex architectures like Attention and Transformers.

FHE-Coder takes a definitive step toward democratizing access to privacy-preserving technologies, enabling developers to generate code that is simultaneously functional, verifiably secure, and reliable.

Contact

mayank.kumar@ucf.edu

Project URL

https://fhe-coder.github.io