COMPREHENSIVE RESOURCE USAGE ANALYSIS OF HOMOMORPHIC ENCRYPTION LIBRARIES IN CLIENT-SERVER ARCHITECTURES

Authors: Vincent Cohadon, Jessy Fallavier, Jiahui Xiang, Osman Salem, Ahmed Mehaoua

Summary

What is Homomorphic Encryption ?

Definition: A cryptographic system that allows computations on encrypted data without decryption, ensuring confidentiality.

Key properties:

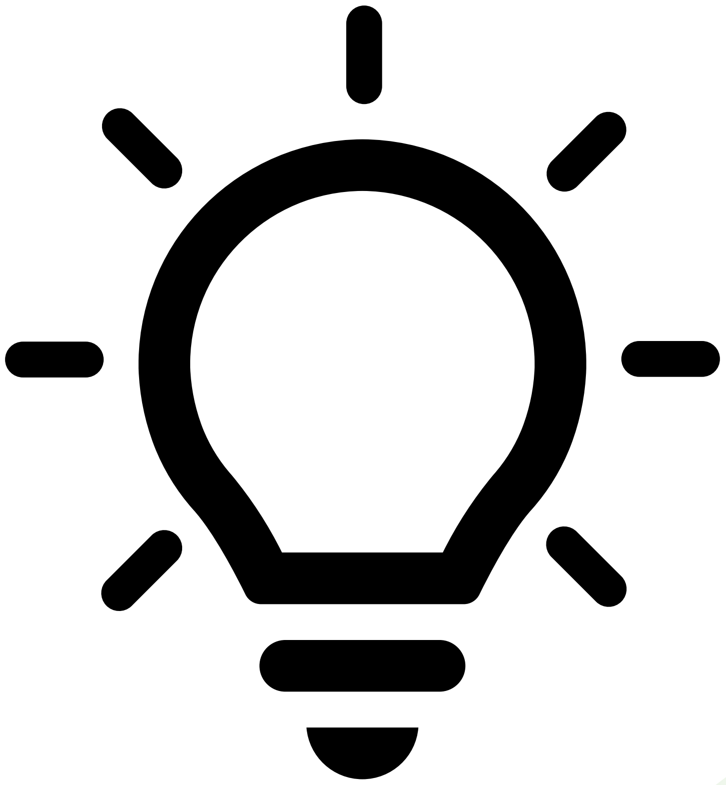
$$Encrypt(a) \oplus Encrypt(b) = Encrypt(a + b)$$
$$Encrypt(a) \otimes Encrypt(b) = Encrypt(a \cdot b)$$

Applications:

- Healthcare analytics
- Cloud computing
- Privacy-preserving machine learning

Historical Context:

- *Introduced by Rivest, Adleman and Dertouzos (1978)*
- *First fully homomorphic scheme by Craig Gentry (2009)*



HE Classification & Capabilities

Type	Operations	Examples	Limitations
Partially (PHE)	Limited (e.g., addition)	Paillier	Single operation type
Somewhat (SHE)	Add + Multiply (limited)	Early BFV	Noise growth limits depth
Fully (FHE)	Unlimited via bootstrapping	BFV, CKKS, TFHE	Higher computational cost

Research Motivation & Gap

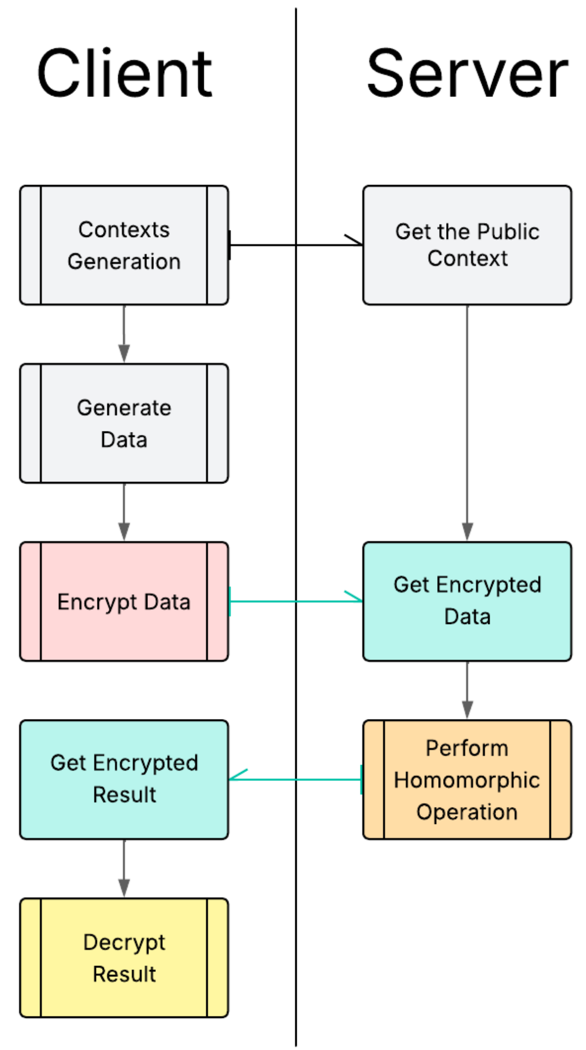
- Challenge: HE operations take seconds/minutes vs. microseconds for plaintext.
- Problem: Ciphertexts 100x-1000x larger than plaintext, straining memory and bandwidth.
- Gap: Lack of practical guidance for HE deployment in client-server systems.
- Our Focus: Empirical resource profiling (CPU, memory, network, disk) for real-world decisions.

Studied HE Schemes

Protocol	Paillier	BFV	CKKS	TFHE
Homomorphic Type	Partially	Somewhat	Approximate	Full
Base	Factoring	RLWE		LWE
Homomorphic Operations	$\oplus$	$\oplus$ & $\otimes$		Boolean circuits
Python Library	Python-Paillier	TenSEAL		Concrete

Experimental Methodology

- **Architecture**: Client-server model with separate VMs
- **Client Role**: Encrypts data, sends to server, decrypts results
- **Server Role**: Performs homomorphic operations (no private context access)
- **Metrics**:
  - Execution time
  - CPU utilization
  - Memory usage
  - Bandwidth
  - Disk I/O
- **Standardization**: Identical hardware, 10x repetition



Python Implementation Benchmark Framework

"Run" class stats

(min, max, sum, avg)

Measurement each 0.1 s

"Aggregated" class stats

(min, max, average of Run)

Interpolation on 20 points

As a Python decorator

Autonomous framework

Results Generation

Plots in png

Profiling results in md

Python Implementation HE Scenario

Abstract class "HEScheme"

Implemented for each scheme

Contexts Generation, Encryption, Serialization, Operations

Parameterized code by arguments

Client / Server mode, IP, schemes, operations, runs

Scenario execution

Socket connection

Context generation

Client and Server behavior

Python Implementation Improvements

- Key exchange excluded of the benchmark
- One-time key generation and socket creation
- Byte counting instead of intermediate virtual machine
- As TFHE use a circuits, we perform the operation on the server circuit with generated evaluation key
- 2 VM are used so the architecture stay the same and results are comparable

System Configuration

Type	Specification	Details
Virtual Machine (Client or Server)	Operating System (VM)	Debian 12 via KVM
	RAM	2 GB
	CPU	2 CPUs
	Storage	20 GB SSD
Host	Memory Type	LPDDR5X
	Performance Core (per core)	Base: 1.4 GHz, Turbo: 4.8 GHz
	Efficient Core (per core)	Base: 0.9 GHz, Turbo: 3.8 GHz

Benchmark Scenarios

- **Scenario 1: PHE vs FHE**
  - 16 ciphertexts, 16 additions
  - Paillier vs. BFV
- **Scenario 2: FHE Comparison**
  - 512 ciphertexts, 32 additions
  - BFV, CKKS, TFHE
  - Key length: 4096 bits
  - Data range: 0 to 2^7 (TFHE)

Key Results
Scenario 1 (Paillier vs BFV)

- **Performance Comparison** (averages):
  - Encryption: Paillier 7.85s, BFV 0.016s (**488x faster**)
  - Operation: Paillier 0.027s, BFV 0.008s (**3.4x faster**)
  - Decryption: Paillier 2.13s, BFV 0.005s (**426x faster**)
- **Insight**: BFV excels for simple addition workloads.

Key Results
Scenario 2 (FHE Comparison)

- **Performance Comparison** (averages):

Phase	BFV	CKKS	TFHE
Encryption	0.44s	0.73s	0.65s
Operation	0.61s	0.88s	13.96s
Decryption	0.13s	0.20s	0.33s

- **Insight**: BFV fastest overall; TFHE slowest due to bootstrapping.

Resource Consumption Analysis

Memory Usage (Scenario 2):

Client: BFV 81.93 MB, CKKS 68.12 MB, TFHE 0.26 MB

Server: BFV 69.68 MB, CKKS 98.44 MB, TFHE 0.61 MB

Bandwidth (Scenario 2):

BFV: 879.54 MB

CKKS: 959.53 MB

TFHE: 50.65 MB (**18x less**)

Bottleneck: Large ciphertexts (~900 MB) for BFV/CKKS.

Practical Guidelines

- **BFV**: Fast integer ops, moderate-depth, ML inference
- **CKKS**: Floating-point, neural networks, statistical analysis
- **TFHE**: Boolean circuits, unlimited depth, low-bandwidth needs
- **Paillier**: Avoid for multi-operation tasks (high overhead)

Live Demonstration

Demo Steps:

- Client-Server VMs
- Python code
- HE Scenario execution
- Plots and Profiling Results

Conclusions & Future Work

- Key Contributions:
  - Comprehensive resource analysis in client-server context
  - Practical guidelines for HE scheme selection
  - Bandwidth identified as major bottleneck
- Future Directions:
  - Support string operations/non-integer data
  - Optimize serialization and communication
  - Explore hybrid HE schemes

Thanks

- Thank you to Mr. Jiahui Xiang for helping and following us along this project
- Thank you Mr. Osman Salem & Mr. Ahmed Mehaoua for support during this year
- Thanks to Mr. Mohamad Ali for the Python course, which was very useful for this project
- Also thank you Borelli Laboratory for allowing us to submit our paper on their behalf

Questions & Answers

Please feel free to ask if you have any questions.

We will be happy to answer them.

References

- A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: theory and implementation," ACM Comput. Surv., vol. 51, no. 4, pp. 1-35, 2018.
- M. M. A. Aziz, M. T. M. Tamal, and N. Mohammed, "Secure genomic string search with parallel homomorphic encryption," Information, vol. 15, no. 1, p. 40, 2024.
- N. Bon, D. Pointcheval, and M. Rivain, "Optimized homomorphic evaluation of Boolean functions," IACR TCHES, vol. 2024, no. 3, pp. 302-341, 2024.
- J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in Proc. Advances in Cryptology – ASIACRYPT 2017: 23rd Int. Conf. Part I, 2017, pp. 409-437.
- I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: Fast Fully Homomorphic Encryption over the Torus," Journal of Cryptology, vol. 33, no. 1, pp. 34-91, 2020.
- J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption," Cryptology ePrint Archive, Report 2012/144, 2012. [Online].
- W. Liu, L. You, Y. Shao, X. Shen, G. Hu, J. Shi, and S. Gao, "From accuracy to approximation: A survey on approximate homomorphic encryption and its applications," Computer Science Review, vol. 55, 2025, Art. no. 100689.
- G. K. Mahato and S. K. Chakraborty, "A comparative review on homomorphic encryption for cloud security," IETE J. Research, vol. 69, no. 8, pp. 5124-5133, 2021.
- K. Munjal and R. Bhatia, "A systematic review of homomorphic encryption and its contributions in healthcare industry," Complex Intell. Syst., vol. 9, no. 3, pp. 3759-3786, 2023.
- A. Nakashima, T. Hayashi, H. Tsuchida, Y. Sugizaki, K. Mori, and T. Nishide, "Faster homomorphic evaluation of arbitrary bivariate integer functions via homomorphic linear transformation," in Proc. 12th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, 2024, pp. 76-86.
- R. Onishi, T. Suzuki, S. Sakai, and H. Yamana, "Security and performance-aware cloud computing with homomorphic encryption and trusted execution environment," in Proc. 12th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, 2024, pp. 36-42.
- P. Paillier, "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," in Advances in Cryptology—EUROCRYPT ’99, J. Stern, Ed., Lecture Notes in Computer Science, vol. 1592, Springer, Berlin, Heidelberg, 1999, pp. 223-238.
- S. Rajashree, B. Vineetha, A. B. Mehta, and P. B. Honnavalli, "Homomorphic encryption approach for string concatenation," in Proc. 4th Int. Conf. on Cybernetics, Cognition and Machine Learning Applications, 2022, pp. 267-272.
- Y. B. Wiryen, N. W. A. Vigny, M. J. Ngono, and F. L. Aimé, "A comparative study of BFV and CKKS schemes to secure IoT data using TenSeal and Pyfhel homomorphic encryption libraries," Int. J. of Smart Security Technologies, vol. 10, no. 1, pp. 1-17, 2024.
- https://github.com/data61/python-paillier
- https://github.com/OpenMined/TenSEAL
- https://github.com/zama-ai/concrete