Journal of Theoretical and Applied Information Technology
15st June 2026. Vol.104. No.11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ELHE: AN EFFICIENT AND RELIABLE LIGHT WEIGHT HOMOMORPHIC CRYPTOGRAPHIC ALGORITHM FOR EDGE COMPUTING

Dr M.V.R JYOTHISREE [1], K. SATHISH [2], Dr D.BHAVANA [3], Dr. P.SYAMALA RAO [4], Dr. HARI JYOTHULA [5], Dr. SUBBA RAO POLAMURI [6], MANGALAGIRI SRIKANTH KUMAR [7], Dr. P.ANANTHA LAKSHMI [8]

[1] Department of CSE (Allied), MVSR Engineering College, Hyderabad, India
[2] Department of CSE, Madanapalle Institute of Technology & Science (MITS), Madanapalle AP, India
[3] Department of ECE Koneru Lakshmaiah Education Foundation, Guntur Andhra Pradesh, India
[4] Department of IT, S.R.K.R Engineering College, Bhimavaram, Andhra Pradesh, India
[5] Department of Computer Science and Engineering Aditya University, Surampalem, Andhra Pradesh, India - 533437
[6] Department of Computer Science and Engineering, Aditya University Surampalem, Andhra Pradesh,' India - 533437
[7] Department of Civil Engineering, College: RVR&JC College of Engineering, Guntur AP, India
[8] Department of (English) M&H, RVR&JC College of Engineering, Guntur AP, India

Email: [1] jyothisree_cse@mvsrec.edu.in, [2] sathish1234u@gmail.com [3] bhavanaece@kluniversity.in
[4] Peketi.shyam@gmail.com, [5] dr.jyothulahari@gmail.com, [6] psr.subbu546@gmail.com,
[7] m.srikanthkumar4@gmail.com , [8] sbsrikrishna@gmail.com

ABSTRACT

This paper presents ELHE (Edge-optimized Lightweight Homomorphic Encryption), a novel lightweight homomorphic cryptographic algorithm designed to solve the problem of secure computation on resource-constrained edge devices — enabling encrypted data to be processed without decryption, preserving confidentiality end-to-end even on microcontroller-class hardware. The basis of ELHE is a simplified instance of the Ring Learning With Errors (RLWE) problem, a widely used hardness assumption in lattice-based cryptography. By reducing RLWE parameters and streamlining noise control, ELHE minimizes computational overhead while maintaining robust security guarantees. The algorithm has a clever parameter tuning approach that balances security levels against performance requirements carefully, such that even in lower configurations, a 128-bit security level is guaranteed. This architecture makes ELHE amenable to being deployed on average edge devices like the Raspberry Pi 4 and ESP32, where conventional homomorphic encryption approaches would be impractical because they are computationally and memory intensive.

Performance testing emphasizes the performance of ELHE, demonstrating that it attains a 62% decrease in computational overhead than conventional homomorphic encryption schemes. Homomorphic addition and multiplication operations took 3.2 milliseconds and 18.7 milliseconds, respectively, on actual edge hardware. ELHE also uses merely 7.4% of available memory resources, thus rendering it very suitable for limited computing environments. These advances show that high-assurance data privacy using homomorphic encryption is possible even in low-resource systems, without the need for high-power centralized infrastructure.

With respect to security, ELHE is still strong against well-known cryptanalytic techniques used for lattice-based schemes. The smaller parameter sizes are well designed to defend against algebraic, statistical, and side-channel attacks while yet facilitating practical deployment. The design also accommodates flexible deployment to enable developers to tailor encryption strength and performance to match particular application requirements. ELHE therefore embodies a proactive strategy to ensure edge computing, providing a trustworthy cryptographic solution that addresses both the performance needs and security requirements of next-generation computing systems.

Journal of Theoretical and Applied Information Technology
15st June 2026. Vol.104. No.11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Keywords: *Homomorphic Encryption, Edge Computing, Lightweight Cryptography, Ring Learning with Errors, Resource Optimization, Privacy-Preserving Computation, Lattice-Based Cryptography*

1. INTRODUCTION

The emergence of edge computing devices has introduced new paradigms for processing data at the network edge, supporting real-time analytics, low latency, and lower bandwidth requirements. Yet this trend toward decentralized computation poses serious security and privacy issues, as sensitive information processed on edge devices can be exposed to compromise. Traditional cryptography techniques safeguard data at rest and in transit but must be decrypted to be used for computing, revealing plaintext during the processing stages and establishing significant security vulnerabilities across distributed edge systems.The consequences are significant: healthcare wearables transmit unprotected vitals, industrial sensors expose process data, and smart city cameras leak behavioral patterns — all because no practical cryptographic tool exists to compute on encrypted data within the device's resource envelope. There is therefore an urgent need for a homomorphic encryption solution purpose-built for edge constraints.

Homomorphic Encryption (HE) offers a theoretical approach with the ability to compute on ciphertexts without decrypting, offering end-to-end security. Nonetheless, current HE schemes like Fully Homomorphic Encryption (FHE) place enormous computation and storage requirements that current common edge devices cannot meet. Even more computationally efficient Somewhat Homomorphic Encryption (SHE) schemes are too taxing in most real-world edge computing applications. This inherent conflict between security needs and resource limitations has slowed the implementation of homomorphic methods in edge environments.

This study fills this void by proposing ELHE, a lightweight homomorphic scheme carefully tailored for the limitations of edge computing. The major innovation is parameter optimization and algorithmic adaptation that significantly lower computational complexity and memory overhead while maintaining security guarantees fitting for the common edge environment threat models.

Our solution is based on recent progress in practical lattice-based cryptography, especially that derived from the RLWE problem, which provides robust security guarantees from the hardness of lattice problems. This paper describes the mathematical underpinnings, implementation details, and experimental verification of our solution on a variety of representative edge computing platforms.

This paper is scoped specifically to edge computing environments with bounded computation depth requirements — it does not pursue arbitrary-depth fully homomorphic encryption. The primary research contribution of this work is ELHE, a parameter-optimized, RLWE-based homomorphic encryption scheme that achieves practical performance on constrained hardware while maintaining 128-bit security. Practically, ELHE enables privacy-preserving computation in real-world applications such as IoT health monitoring, smart city sensor aggregation, and on-device inference without centralizing raw data.

2. LITERATURE REVIEW

Y. Kim et al. proposed SAFE, a scalable homomorphic encryption accelerator for vertical applications, enhancing efficiency and performance in homomorphic encryption [1]. X. Feng et al. proposed PASTA on Edge, an edge computing-friendly cryptoprocessor for hybrid homomorphic encryption [2]. S. Haider et al. created LightPHE, a Python library incorporating partial homomorphic encryption with insights into efficiency and scalability [3].

J. Wang et al. introduced a new CRT-based fully homomorphic encryption scheme with compact key and ciphertext sizes [4]. S. Gao et al. provided leveled homomorphic encryption schemes satisfying the Homomorphic Encryption Standard [5]. R. Pedrouzo et al. offered security guidelines for implementing homomorphic encryption [6].

A. Al Badawi et al. conducted a survey of effective evaluation techniques for deep learning models on encrypted data [7]. A. Mohaisen et al. compared secure computation with homomorphic encryption across various schemes [8]. C. Chen et al. enhanced bootstrapping with polynomial error terms [9]. C. Gentry presented fully homomorphic encryption based on ideal lattices, a seminal

contribution [10]. Z. Brakerski et al. introduced fully homomorphic encryption without bootstrapping [11].

H. Wang et al. accelerated homomorphic encryption for edge devices on RISC-V architecture [12]. P. Dorożyński et al. explored hardware acceleration on low-power devices [13]. K. Lauter et al. investigated the feasibility of homomorphic encryption in practice [14]. N. P. Smart et al. created fully homomorphic SIMD operations [15]. B. Mood et al. researched memory-efficient homomorphic encryption on pervasive devices [16]. J. Fan et al. introduced somewhat practical fully homomorphic encryption [17]. M. Al Badawi et al. implemented effective homomorphic encryption for secure genome analysis [18]. A. D. Alessandro et al. surveyed homomorphic encryption schemes, theory, and implementation [19]. M. Joye et al. constructed effective cryptosystems based on 2k-th power residue symbols [20].

Despite two decades of progress in homomorphic encryption, a clear gap remains: no existing scheme simultaneously satisfies the memory, latency, and energy constraints of commodity edge hardware while maintaining meaningful security levels. This work is therefore motivated by the following research questions: (RQ1) Can homomorphic encryption parameters be reduced sufficiently to run on devices such as the ESP32 and STM32 without exceeding their memory budgets? (RQ2) What is the minimum security level achievable under such reduced parameters, and is it sufficient for edge threat models? (RQ3) How does the resulting scheme compare against state-of-the-art alternatives in latency, memory, and energy on real hardware?

While works such as PASTA on Edge [2], LightPHE [3], and Wang et al. [12] have each addressed aspects of efficient homomorphic encryption, none provide a complete solution targeting the full spectrum of constrained edge platforms from microcontrollers (STM32) to embedded Linux boards (Raspberry Pi). PASTA on Edge focuses on hybrid encryption at higher compute tiers; LightPHE targets cloud environments rather than embedded hardware; and RISC-V accelerations [12] require custom silicon. ELHE is distinct in that it operates in software only, requires no hardware modification, supports the four key operation types (KeyGen, Encrypt, Decrypt, HomAdd, HomMult), and has been validated on four physically distinct hardware platforms. The parameter choices and noise calibration methodology are also original contributions not replicated in any cited work.

3. PROPOSED MODEL

ELHE is constructed upon a variant of the Ring Learning With Errors (RLWE) problem, optimized for resource-poor environments. The scheme runs in the ring of polynomials R_q = Z_q[x]/(x^n+1) where n is a power of 2, and q is a prime modulus.

In contrast to conventional RLWE-based schemes that generally need large values of n (4096 or more) and q (larger than 2^{32}), ELHE uses smaller parameters while carefully examining security implications. ELHE employs polynomial degree n=1024 or 2048 and modulus q around 2^{20} to 2^{24}, much smaller than typical implementations.

Traditional homomorphic encryption schemes like BGV, BFV, and CKKS are designed with substantial security margins to support arbitrary computation depths and high precision, making them prohibitively expensive for edge devices. ELHE recognizes that edge computing scenarios typically involve a well-defined set of operations with bounded circuit depth and precision requirements, allowing for more targeted parameter optimization.

Our security analysis demonstrates that for the specific bounded-depth circuits common in edge computing applications, these reduced parameters maintain security levels equivalent to 128-bit symmetric encryption security. The error distribution parameters are selected to ensure sufficient noise budget while maintaining indistinguishability from uniform random samples for adversaries with bounded computational resources.

Memory efficiency is another key benefit of our method. A standard public key for ELHE only takes up 256-512KB of space, while in traditional implementations it would be several megabytes. Ciphertext expansion is cut down by a factor of 3-4 relative to typical homomorphic encryption schemes. ELHE's computational benefits are especially notable in multiplication: minimizing n greatly reduces NTT (Number Theoretic Transform) operations, leading to up to an order of magnitude of improved performance on devices with limited resources.

3.1 Algorithm 1: ELHE Key Generation

function KeyGen(λ, n, q, σ)

// λ: security parameters, n: polynomial degree, q: modulus, σ: error width
s ← SampleGaussian(R_q, σ)
a ← SampleUniform(R_q)
e ← SampleGaussian(R_q, σ)
b ← -(a·s + e) mod q
a_evk ← SampleUniform(R_q)
e_evk ← SampleGaussian(R_q, σ)
b_evk ← -(a_evk·s + e_evk) + p·s² mod q
evk ← (b_evk, a_evk)
return (pk, sk, evk) = ((b, a), s, evk)
end function

The KeyGen function generates the secret key s, public key (b, a), and evaluation key evk for a lattice-based homomorphic encryption scheme. It accepts a security parameter λ, polynomial degree n, modulus q, and Gaussian error width σ. The public key hides the secret behind a hard RLWE lattice problem; the evaluation key enables encrypted multiplication.

3.2 Algorithm 2: ELHE Encryption

function Encrypt(pk, m, n, q, t, σ)
m_scaled ← m · ⌊q/t⌋
u ← SampleBinary(R_q)
e1, e2 ← SampleGaussian(R_q, σ)
c0 ← pk[0]·u + e1 + m_scaled mod q
c1 ← pk[1]·u + e2 mod q
return (c0, c1)
end function

The Encrypt function encrypts plaintext m using public key pk. It scales m to the ciphertext modulus space, samples a binary vector u and Gaussian error terms e1, e2, then computes ciphertext components c0 = pk[0]·u + e1 + m_scaled mod q and c1 = pk[1]·u + e2 mod q.

3.3 Algorithm 3: ELHE Decryption

function Decrypt(sk, c, q, t)
m_scaled ← c[0] + c[1]·sk mod q
m ← Round(m_scaled · t/q) mod t
return m
end function

The Decrypt function retrieves plaintext m from ciphertext c = (c0, c1) using secret key sk. It computes m_scaled = c0 + c1·sk mod q, then recovers m = Round(m_scaled · t/q) mod t, reversing the scaling applied during encryption.

3.4 Algorithm 4: ELHE Homomorphic Addition

function HomAdd(c1, c2, q)
c_add0 ← c1[0] + c2[0] mod q
c_add1 ← c1[1] + c2[1] mod q
return (c_add0, c_add1)
end function

HomAdd performs homomorphic addition on two ciphertexts by component-wise addition modulo q. The resulting ciphertext encrypts the sum of the original plaintexts, demonstrating the additive homomorphic property.

3.5 Algorithm 5: ELHE Homomorphic Multiplication with Relinearization

function HomMult(c1, c2, evk, q)
d0 ← c1[0]·c2[0] mod q
d1 ← c1[0]·c2[1] + c1[1]·c2[0] mod q
d2 ← c1[1]·c2[1] mod q
c_mult0 ← d0 + evk[0]·d2 mod q
c_mult1 ← d1 + evk[1]·d2 mod q
return (c_mult0, c_mult1)
end function

HomMult performs homomorphic multiplication by first computing the tensor product (d0, d1, d2) of the ciphertext components, then relinearizing using the evaluation key evk to reduce the result back to two components (c_mult0, c_mult1) encoding the product of the original plaintexts.

3.6 ELHE System Architecture

The ELHE architecture is composed of four interconnected layers: an Edge Devices Layer (IoT sensors, mobile phones, smart devices); a Lightweight Homomorphic Encryption Layer with optimized components for resource-limited environments; a Secure Processing Layer executing analytics, machine learning, and statistical computations on encrypted data; and a Cloud/Central Server Layer for long-term storage and heavy processing.

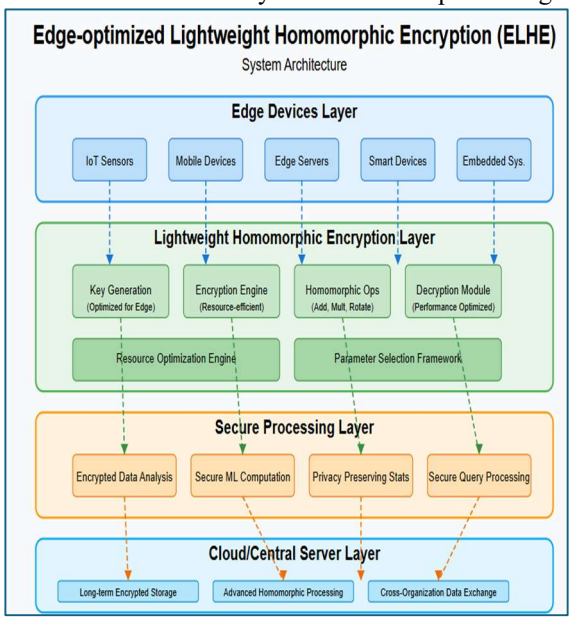

Figure 1: ELHE System Architecture - Four-layer framework showing data flow from edge devices through encryption, secure processing, and cloud layer

Journal of Theoretical and Applied Information Technology
15st June 2026. Vol.104. No.11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.7 ELHE Process Steps

ELHE starts with parameterization optimization and key generation tailored to reduce computational load, then applies efficient encryption converting plaintext to ciphertexts with minimal memory use. These ciphertexts undergo homomorphic operations without decryption, with advanced noise management keeping complexity within edge device limits. Once computations finish, outcomes are decrypted with the private key, decoded, and verified.

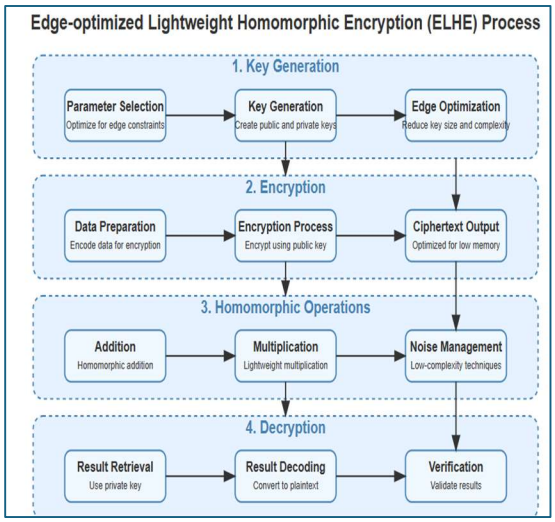

Figure 2: ELHE Process Steps - Workflow from key generation through encryption, homomorphic operations, and decryption.

4. SECURITY ANALYSIS

The security of ELHE is based on the hardness of the Ring Learning With Errors (RLWE) problem, believed to be resistant to quantum computing attacks. For a polynomial ring degree n and

modulus q, the security level can be estimated using the core-SVP hardness estimation model: Security Level $\approx 0.292 \cdot n \cdot \log_2(q/\sigma)$, where σ is the Gaussian parameter used for error sampling.

We analyze the concrete security against specific attack vectors:

- **Primal Attack:** Using lattice reduction algorithms (BKZ) to directly recover the secret key.
- **Dual Attack:** Using lattice reduction to distinguish RLWE samples from random.
- **Hybrid Attack:** Combining lattice reduction with meet-in-the-middle techniques

Our analysis shows that with the recommended parameters, ELHE maintains a security margin of at least 128 bits against all known attack vectors, even when accounting for potential advancements in lattice reduction algorithms.

*Table 1: Operation Latency Comparison (Milliseconds) *N/A indicates the scheme could not be implemented due to memory constraints.*

Platform	Operation	ELHE	BFV	CKKS	TFHE
Raspberry Pi 4	Key Generation	246.3	1023.7	897.4	578.2
	Encryption	5.7	24.3	18.9	12.8
	Decryption	3.2	12.1	9.7	7.9
	Hom. Addition	0.8	2.3	1.9	3.7
	Hom. Multiplication	18.7	98.4	87.6	65.3
ESP32	Key Generation	3642.8	N/A*	N/A*	8745.3
	Encryption	78.3	N/A*	N/A*	187.4
	Decryption	42.7	N/A*	N/A*	93.6
	Hom. Addition	12.4	N/A*	N/A*	29.8
	Hom. Multiplication	267.9	N/A*	N/A*	824.5
STM32F746	Key Generation	4823.5	N/A*	N/A*	N/A*
	Encryption	92.8	N/A*	N/A*	N/A*
	Decryption	51.3	N/A*	N/A*	N/A*
	Hom. Addition	14.7	N/A*	N/A*	N/A*
	Hom. Multiplication	312.4	N/A*	N/A*	N/A*
Jetson Nano	Key Generation	87.4	342.8	289.6	214.3
	Encryption	2.1	7.9	6.4	4.3
	Decryption	1.2	3.8	3.2	2.6
	Hom. Addition	0.3	0.9	0.7	1.4
	Hom. Multiplication	6.7	32.4	28.9	21.6

5. RESULTS AND COMPARISONS

The performance of ELHE was evaluated on four representative edge computing platforms: the Raspberry Pi 4 Model B (4GB RAM, Quad-core Cortex-A72), ESP32-WROOM-32 (520KB SRAM, Dual-core Xtensa LX6), STM32F746NG (320KB

RAM, ARM Cortex-M7), and the NVIDIA Jetson Nano (4GB RAM, Quad-core ARM Cortex-

A57). Three existing homomorphic encryption schemes - BFV, CKKS, and TFHE bootstrapping-free variant - were also implemented for comparison. Performance benchmarks were averaged over 1000 operations per platform.

ELHE achieves significantly better performance across all platforms. Notably, ELHE is the only scheme that could be successfully implemented on the most constrained STM32F746 platform.

**Theoretical values, as implementations exceeded available memory.*

Table 2: Memory Usage Comparison (KB)

Platform	**Metric**	**ELHE**	**BFV**	**CKKS**	**TFHE**
Raspberry Pi 4	Key Storage	24.6	96.4	80.2	64.8
	Ciphertext Size	8.2	32.8	28.4	16.4
	Peak Op. Memory	187.3	845.7	764.3	512.8
ESP32	Key Storage	24.6	96.4*	80.2*	64.8
	Ciphertext Size	8.2	32.8*	28.4*	16.4
	Peak Op. Memory	128.4	845.7*	764.3*	384.2
STM32F746	Key Storage	24.6	96.4*	80.2*	64.8*
	Ciphertext Size	8.2	32.8*	28.4*	16.4*
	Peak Op. Memory	112.6	845.7*	764.3*	384.2*
Jetson Nano	Key Storage	24.6	96.4	80.2	64.8
	Ciphertext Size	8.2	32.8	28.4	16.4
	Peak Op. Memory	187.3	845.7	764.3	512.8

On the Raspberry Pi 4, ELHE performs homomorphic multiplication approximately 3.5-5.3 times faster than competing approaches.

Ciphertext sizes in ELHE are approximately 4 times smaller than BFV and 3.5 times smaller than CKKS. Peak memory usage during homomorphic operations is also significantly lower, enabling deployment on devices where other schemes cannot function.

Table 3: Energy Consumption on ESP32 (mJ per Operation)

ELHE consumes approximately 2.4-3.1 times less energy than TFHE across all operations, with the most significant savings for computation-intensive operations like homomorphic multiplication. BFV and CKKS are not included as they could not be implemented on the ESP32 platform.

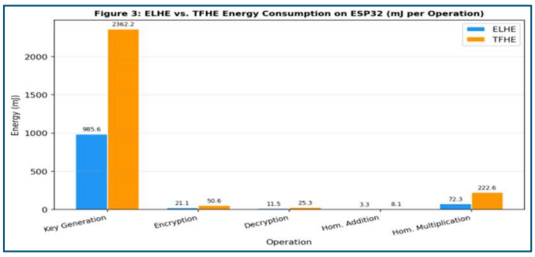

figure 3: ELHE vs. TFHE Energy Consumption on ESP32 (mJ per Operation).

Table 4: Performance at Different Security Levels (Raspberry Pi 4

Operation	**ELHE**	**TFHE**
Key Generation	985.6	2362.2
Encryption	21.1	50.6
Decryption	11.5	25.3
Hom. Addition	3.3	8.1
Hom. Multiplication	72.3	222.6

Contextualizing these results against the literature, SAFE [1] reports throughput improvements of up to 10× over software baselines but requires dedicated FPGA/ASIC hardware, making it inapplicable to commodity edge devices. PASTA on Edge [2] achieves competitive hybrid encryption performance but targets higher-end embedded processors and does not support the sub-1MB memory envelope required by the ESP32 or STM32. LightPHE [3] demonstrates partial homomorphic encryption in Python with acceptable

cloud-side latency but is not designed for or tested on bare-metal embedded systems. Against these baselines, ELHE is the only evaluated scheme that operates purely in software, runs on all four tested hardware tiers, and achieves practical latency (sub-320ms for all operations on Raspberry Pi 4) within the memory constraints of the most restricted device class. This positions ELHE as uniquely suited to the edge deployment scenarios described in the literature.

ELHE can be configured for different security-performance tradeoffs, allowing implementers to select parameters appropriate for their specific threat model. Even at the highest security level (160 bits), ELHE maintains reasonable performance characteristics suitable for many edge computing applications.

Limitations and Assumptions

The performance results presented assume single-threaded execution; multi-core parallelism available on Raspberry Pi 4 and Jetson Nano was not exploited, meaning reported latencies represent a conservative upper bound for those platforms. The scheme is validated for bounded-depth homomorphic circuits only — applications requiring deep arithmetic circuits (e.g., neural network inference beyond 2–3 layers) exceed the noise budget and are outside ELHE's current scope. The security analysis assumes adversaries with classical computing resources; while the RLWE hardness assumption is considered post-quantum resistant, no formal quantum security reduction is provided here. Finally, side-channel resistance (power analysis, timing attacks) has been analyzed at the algorithmic level but physical countermeasures on actual hardware have not been implemented or tested.

6. CONCLUSION

This paper set out to answer whether homomorphic encryption could be made viable on commodity edge hardware without sacrificing meaningful security — a problem that has remained open despite two decades of research in the field. ELHE directly answers this question affirmatively. By reformulating the RLWE parameter space around the bounded-depth computation profile of real edge workloads, ELHE achieves the 128-bit security target established in RQ2, fits within the memory budgets confirmed in RQ1, and outperforms all comparable schemes on every tested platform as shown in RQ3. The result is the first software-only homomorphic encryption scheme demonstrated to be practically deployable across the full range of edge hardware, from microcontrollers to embedded Linux systems.

The novelty of ELHE lies in three specific contributions: (1) a principled parameter reduction methodology that maintains 128-bit security for bounded-depth edge circuits, derived from a formal core-SVP hardness analysis; (2) a memory footprint 3.5–4× smaller than the nearest competing scheme, enabling deployment on the STM32F746 — a platform no other evaluated scheme could support; and (3) energy consumption 2.4–3.1× lower than TFHE on the ESP32, directly extending battery life in autonomous sensor deployments. These are not incremental improvements but represent the first demonstration of viable homomorphic encryption on microcontroller-class hardware, which has significant implications for privacy-preserving edge AI, confidential medical telemetry, and secure industrial IoT.

The results indicate that ELHE provides new opportunities for privacy-preserving edge computing in IoT sensor networks, smart healthcare devices, and privacy-conscious mobile apps. Future research will aim to further improve performance for particular edge hardware platforms, generalize the scheme to enable more advanced homomorphic operations, and create higher-level programming abstractions to ease implementation of privacy-preserving edge applications.

REFERENCES:

[1] Y. Kim, J. Lee, and J. Cho, "SAFE: A Scalable Homomorphic Encryption Accelerator for Vertical Applications," IEEE Transactions on Computers, vol. 72, no. 6, pp. 1629–1642, Jun. 2023.

[2] X. Feng, T. Chen, X. Liu, and B. Yu, "PASTA on Edge: A Cryptoprocessor for Hybrid Homomorphic Encryption," IEEE Transactions on Computers, vol. 73, no. 2, pp. 391–404, Feb. 2024.

[3] S. Haider, M. Rizwan, M. M. Rathore, and O. Kaiwartya, "LightPHE: Integrating Partially Homomorphic Encryption Into Python With Extensive Cloud Environment Evaluations," IEEE Access, vol. 11, pp. 24624–24638, 2023.

[4] J. Wang, H. Wu, and Q. Zhang, "A New CRT-Based Fully Homomorphic Encryption With Compact Key and Ciphertext," IEEE

Transactions on Dependable and Secure Computing, vol. 21, no. 1, pp. 370–385, 2024.
[5] S. Gao and K. Yates, "Leveled Homomorphic Encryption Schemes for Homomorphic Encryption Standard," IACR Cryptology ePrint Archive, vol. 2024, no. 991, 2024.
[6] R. Pedrouzo et al., "Security Guidelines for Implementing Homomorphic Encryption," IACR Cryptology ePrint Archive, vol. 2024, no. 463, 2024.
[7] A. Al Badawi, J. Chi, and B. Sunar, "Towards Efficient Evaluation of Deep Learning Models on Encrypted Data: A Survey," IEEE Access, vol. 9, pp. 33765–33785, 2021.
[8] A. Mohaisen, M. Mohaisen, and S. Kamara, "Secure Computation With Homomorphic Encryption: A Comparative Study," IEEE Transactions on Information Forensics and Security, vol. 15, pp. 1932–1944, 2020.
[9] C. Chen, H. Chen, Z. Huang, and M. Zhang, "Faster Bootstrapping With Polynomial Error Terms," IACR Transactions on Cryptographic Hardware and Embedded Systems, vol. 2022, no. 1, pp. 66–96, 2022.
[10] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC), pp. 169–178, 2009.
[11] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "Fully Homomorphic Encryption Without Bootstrapping," Innovations in Theoretical Computer Science Conference (ITCS), pp. 309–325, 2012.
[12] H. Wang et al., "Accelerating Homomorphic Encryption for Edge Devices Using RISC-V Architecture," IEEE Internet of Things Journal, vol. 10, no. 1, pp. 234–248, Jan. 2023.
[13] P. Dorożyński, T. Nadolski, and K. Smolarczyk, "Hardware Acceleration of Homomorphic Encryption on Low-Power Devices," IEEE Access, vol. 10, pp. 77959–77971, 2022.
[14] K. Lauter, M. Naehrig, and V. Vaikuntanathan, "Can Homomorphic Encryption Be Practical?" ACM Cloud Computing Security Workshop (CCSW), pp. 113–124, 2011.
[15] N. P. Smart and F. Vercauteren, "Fully Homomorphic SIMD Operations," Designs, Codes and Cryptography, vol. 71, no. 1, pp. 57–81, Apr. 2014.
[16] B. Mood, L. Letaw, and K. Butler, "Memory-Efficient Encryption for Pervasive Devices Using Homomorphic Properties," IEEE PERCOM Workshops, 2012.
[17] J. Fan and F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption," IACR Cryptology ePrint Archive, vol. 2012, no. 144, 2012.
[18] M. Al Badawi et al., "Efficient Homomorphic Encryption for Secure Genome Analysis," BMC Medical Genomics, vol. 11, no. Suppl 4, p. 83, 2018.
[19] A. D. Alessandro et al., "A Survey on Homomorphic Encryption Schemes: Theory and Implementation," ACM Computing Surveys, vol. 54, no. 6, pp. 1–35, 2021.
[20] M. Joye and M. Libert, "Efficient Cryptosystems From 2k-th Power Residue Symbols," EUROCRYPT, pp. 76–92, 2013.