Annals of Optimization With Applications

www.anowa.reapress.com

Ann. Optim. Appl. Vol. 1, No. 4 (2026) 269–287.

Paper Type: Original Article

Blind Computation in AI Machine Operation: A Study of Homomorphic Encryption, Secure Multi-Party Computation, and Federated Learning

Ikechukwu Bismark Owunna[1,*], Imoh Ime Ekanem[2], Aniekan Essienubong Ikpe[2]

[1] Department of Mechanical Engineering, University of Benin, Benin City, PMB. 1154, Nigeria; Ikechukwu.owunna@eng.uniben.edu.

[2] Department of Mechanical Engineering, Akwa Ibom State Polytechnic, Ikot Osurua, IkotEkpene, Nigeria; Imoh.ekanem@akwaibompoly.edu.ng; Aniekan.ikpe@akwaibompoly.edu.ng.

Citation:

Received: 25 January 2025 *Revised: 26 April 2025* *Accepted: 17 June 2025*	Bismark Owunna, I., Ime Ekanem, I., & Essienubong Ikpe, A., (2026). Blind computation in AI machine operation: A study of homomorphic encryption, secure multi-party computation, and federated learning. *Annals of optimization with applications, 1*(4), 269-287.

Abstract

This study critically examines the emerging field of Blind Computation (BC) in Artificial Intelligence (AI) machine operations, challenging conventional approaches to data privacy and security in artificial intelligence systems. As AI continues to permeate various sectors, from healthcare to finance, the need for robust privacy-preserving techniques has become paramount. BC offers a promising solution by enabling AI models to process encrypted data without decryption, thus maintaining data confidentiality throughout the computational pipeline. The study explored conventional techniques in Homomorphic Encryption (HE), Secure Multi-Party Computation (SMPC), and Federated Learning (FL), presenting a comprehensive framework for implementing BC in AI systems. We propose novel architectures that significantly enhance data protection without compromising computational efficiency. The analysis revealed that while BC techniques offer unprecedented levels of privacy, they also introduce new challenges in terms of computational overhead and model accuracy. Empirical evidence demonstrating the trade-offs between privacy, performance, and precision, and propose innovative strategies to optimize these competing factors. Furthermore, we critically assess the ethical implications of BC , examining its potential to either mitigate or exacerbate existing biases in AI systems. This study was concluded by outlining a roadmap for future research, emphasizing the need for interdisciplinary collaboration to address the technical, ethical, and regulatory challenges associated with BC in AI. The findings obtained have significant implications on the design and deployment of privacy-preserving AI systems across various domains, potentially revolutionizing the way sensitive data is processed in the age of artificial intelligence.

Keywords: Blind computation, Artificial intelligence machine operation, Homomorphic encryption, Secure multi-party computation.

1 | Introduction

The genesis of this research lies at the intersection of mechatronics and cryptography. As described by Bolton [1] in his seminal work on mechatronic design, traditional mechatronic systems have evolved from simple electromechanical devices to complex, Artificial Intelligence (AI)-driven machines. This evolution, while

Corresponding Author: Ikechukwu.owunna@eng.uniben.edu

https://doi.org/10.48314/anowa.v1i4.59

dramatically enhancing Capabilities, has also exponentially increased the volume and sensitivity of data processed by these systems.

Concurrently, the field of cryptography has seen revolutionary advancements. The concept of privacy homomorphisms, introduced by Rivest et al. [2], laid the groundwork for what would become Fully Homomorphic Encryption (FHE). These cryptographic techniques offer a potential solution to the privacy challenges faced by modern mechatronic systems. In the realm of mechatronics, the integration of AI has been transformative. From the precise control systems in manufacturing robots, as outlined by Koren et al. [3], to the complex perception and decision-making algorithms in autonomous vehicles described by Pendleton et al. [4], AI has become an indispensable component of advanced mechatronic systems. However, this integration raises critical questions about data security and privacy, especially when these systems operate in sensitive environments or process personal data. The convergence of cryptographic techniques and AI in mechatronic systems forms the foundation of Blind Computation (BC).

This multifaceted approach encompasses FHE, Secure Multi-Party Computation (SMPC), differential privacy, and zero-knowledge proofs. This framework aims to enable mechatronic systems to operate on encrypted data, performing computations "blindly" without exposing the underlying information. Users have been strongly motivated to do computing on hardware they do not personally control for practically as long as programmable computers have been around. This was initially brought on by the expensive nature of these devices and the requirement for specialized facilities to hold them. Computers were kept in central places by universities, governments, and big businesses, where they executed tasks in batches for their users [5]. Despite the increasing ubiquity of computers, there is still a need for centralized resources. Delegated computation is still widely used nowadays in the form of cloud computing. Even if the future of computing is yet unknown, it makes sense to assume that it will go in a similar direction. Indeed, initiatives to make basic quantum processors accessible over the Internet have given rise to some of this conjecture. Because classical processors are so common and high-speed global communications networks exist, we are in a considerably better position now to permit remote access to computers than we were with early conventional computers. Although there may be compelling practical and financial reasons to outsource computations to distant systems, doing so creates numerous security issues.

Specifically, if the calculation is carried out on unreliable hardware, this can create the potential that either the computation's integrity or privacy could be compromised. Communication can be concealed by encryption between the client and the server from eavesdroppers, while authentication codes are employed to identify any attempt to change these messages. However, such tactics do nothing to neutralize the threat provided by a compromised or malevolent server. Ideally, there would be a method to allay these worries to assign work to a distant server while maintaining confidentiality [6]. Several techniques have surfaced in recent years to address the privacy concerns brought up by delegated computation. Falling under the general category of BC, this gives a client the ability to carry out large processing with one or more distant servers while keeping the structure of the computation concealed. Although BC protocols aim to protect the privacy of only the calculation, numerous additionally permit verification of the computation being carried out, by incorporating hidden tests within the data.

Moreover, findings in traditional secure computing establish a connection between blind computing and computational intricacy. The presence of an adequately safe blind computing protocol using a client that is strictly classical and a one quantum server with the ability to execute any quantum [7]. These computations would draw a link between the concerns of whether BC Protocols comprise NP and whether the polynomial hierarchy collapses. Due to these obstacles, it has only been relatively recently that technologies that could enable this kind of functionality have started to surface.

BC is a broad method that includes safe SMPC, differential privacy, zero-knowledge proofs, and FHE. It is based on the convergence of cryptography and AI in mechatronic systems. In the world of classical cryptography, the state of the art regarding computation on encrypted data has changed dramatically over the years since the introduction of the first secure computation protocols, with the advent of FHE [8], [9]. These

cryptographic methods present a viable remedy for the privacy issues contemporary mechatronic systems are facing. These encryption techniques enable computation on encrypted data without requiring correspondence between both the server and the user while processing. Reaching a decrease in round complexity when using delegated. The use of BC is very desirable. It ought to be observed, though, that even in the traditional setting, this has only been accomplished by utilizing presumptions regarding the hardness of certain computational problems, such as the approximate shortest vector problem, or the learning with errors issue. FHE schemes allow data to be processed arbitrarily in its encrypted form, without the need for the encryption key [10], [11].

Several works have explored the use of Partially-Homomorphic Encryption (PHE) which supports models of computation not classically simulable, including the Boson sampling model, under weakened information-theoretic security guarantees Broadbent and Jeffery introduced a Homomorphic Encryption (HE) scheme that leveraged a computationally secure classical HE scheme to enable the evaluation of circuits containing only a constant number of non-Clifford gates on encrypted quantum data. In the case of Broadbent and Jeffery's scheme, the size of the encoding scaled exponentially with the number of non-Clifford gates. However subsequent work by Dulek et al. [12] reduced this to only polynomial overhead, resulting in a computationally secure leveled HE scheme. These latter protocols can be seen as part of a wider program to broaden quantum cryptographic techniques through the incorporation of computational assumptions.

A recent proposal achieves similar functionality to the Broadbent-Jeffery scheme but under an information theoretic security definition, provided a sufficiently large key is used. To date, however, no such counterpart to the work of Dulek et al. [12] has been found. This study concentrates on theoretical constructions for delegated computation protocols. If these procedures are to evolve into anything more than academic purpose, it is important to develop a road to physical realization. Toward this goal, many attempts have been made to create verifiable blind computing protocols that are easier to test. This has assumed multiple shapes. Although the first suggestions stated that their constructions might be rendered fault-tolerant, significant work has since been put into computing explicit fault-tolerance thresholds for BC and dealing with the issue of noise occurring during communication between user and server.

2 | Research Methodology

2.1 | Preserving the Artificial Intelligence with Secure Computation

I. Conceptual framework development: establishing the theoretical principles guiding BC, incorporating principles from cryptography, AI, and mechatronics.

II. System prototyping and simulation: implementing HE and FL techniques to test their viability in BC.

III. Iterative refinement: continuous improvements based on performance testing, security evaluations, and accuracy benchmarks.

IV. Threat modeling and compliance analysis: identifying potential security vulnerabilities and ensuring adherence to data protection laws such as GDP.

The primary goal is to integrate secure computation techniques into standard AI processing pipelines to enable privacy-preserving machine operations. The methodology is built on an experimental framework, an iterative development process, a theoretical foundation, and stringent security considerations, with a focus on innovation, security, compliance, and practicality. The initial phase entails thoroughly understanding the landscape of privacy-preserving AI approaches [13]. A thorough assessment of academic literature, particularly improvements since 2017, identifies cutting-edge approaches, constraints, and prospective areas for innovation. Key research areas include HE, SMPC, differential privacy, and federate. Secure model aggregation in FL via SMPC is given by *Eq. (1)*, while additive secret sharing reconstruction in SMPC for BC is given by *Eq. (2)*,

$$w_{global}^{t+1} = \frac{1}{N}\sum_{i=1}^{N} \backslash share^{-1}(\backslash share(w_i^{t+1})) \quad (1)$$

$$s = \sum_{i=1}^{k} s_i \pmod p \quad (2)$$

The process then moves on to the conceptual design phase, where it focuses on developing a theoretical model that uses HE to enable BC. This phase entails selecting the optimal HE scheme based on computing complexity, security level, and compatibility with the AI model and data kinds. Potential systems being investigated include BFV, Cheon-Kim-Kim-Song (CKKS), and TFHE. Integral is the creation of a strong data encryption and preprocessing approach that ensures input data is encrypted while remaining useful for AI model training and inference learning. A technology assessment examines available safe compute libraries, frameworks, and hardware accelerators that are critical for easy integration into AI pipelines, based on performance, security, integration, and community support [14]. To improve performance and accuracy, techniques such as data scaling, encoding, and the use of controlled noise are investigated. Simultaneously, the AI model architecture and associated training procedures are modified to accommodate encrypted data, which may necessitate HE-friendly activation functions, optimized linear algebra operations, and specialized training algorithms for encrypted datasets.

Finally, a theoretical performance analysis evaluates the computational cost caused by HE, taking into account encryption/decryption timings, ciphertext sizes, and computational complexity in HE operations. The subsequent stage involves simulation and prototyping. An experimental prototype is developed to integrate secure computation techniques into a realistic AI application (example: medical image analysis with patient data). An iterative development cycle is employed, which is characterized by the cyclical process of prototype implementation, experimental setup, performance testing, accuracy evaluation, security analysis, and iterative refinement. This iterative process begins with the implementation of the proposed BC model using selected HE schemes, AI frameworks (TensorFlow, PyTorch), and secure computation libraries (SEAL, PySyft). Subsequently, a controlled experimental environment is designed to evaluate the performance, accuracy, and security of the prototype [15]. This involves defining appropriate datasets, relevant evaluation metrics (F1 Score), and comprehensive benchmark scenarios.

Following this, rigorous performance testing is conducted to quantitatively measure the computational overhead introduced by HE, including encryption/decryption times, training/inference times, memory usage, and communication bandwidth. Simultaneously, the accuracy of the AI model is evaluated when operating on encrypted data, and compared to the accuracy when operating on plaintext data. Statistical hypothesis testing is employed to determine the significance of any observed accuracy degradation. Critically, a detailed security analysis of the prototype is performed, considering potential attack vectors such as ciphertext manipulation, key leakage, and side-channel attacks [16]. Security analysis tools and techniques, such as fuzzing and formal verification, are utilized to identify vulnerabilities. The prototype is then iteratively refined based on the results of performance testing, accuracy evaluation, and security analysis, potentially involving adjusting HE parameters, optimizing the AI model architecture, or implementing additional security countermeasures. Throughout each iteration, documentation is maintained, capturing any deviations, improvements, and potential pitfalls encountered during the integration of secure computation techniques into conventional AI workflows.

2.2 | System Architecture

The system architecture is designed as a modular, multi-layered framework that integrates secure computation techniques with conventional AI processing pipelines. The design emphasizes compartmentalization, where each module performs a specific function and interfaces with others through defined protocols [17]. This approach maintains data integrity and confidentiality throughout the process, aligning with secure AI system architecture (see *Fig. 1*) principles Google's Secure Artificial Intelligence Framework (SAIF) also categorizes

AI development into areas like data, infrastructure, model, and application, which informs the design of this system. The data acquisition and preprocessing module collects data from numerous sources while ensuring its integrity and quality. To remove noise and outliers, rigorous cleaning and normalization methods are used. Before entering the computational process, the data is first encrypted to ensure its security. These approaches are inspired by modern techniques in privacy-preserving deep learning, which ensure that data preprocessing does not jeopardize future encrypted calculations [9], [18]. This is consistent with secure coding methods and highlights the quality of the algorithms underpinning AI features. The secure computation module is the foundation of the architecture. It executes computations directly on encrypted data without the need for decryption, ensuring input data confidentiality. This module makes use of advanced HE techniques, which are made possible by libraries such as Microsoft SEAL, as well as optimized algorithms for efficient ciphertext arithmetic operations. Detailed error-correction methods and performance-enhancing routines reduce the computing burden associated with encryption-based operations. The design of this module is based on cutting-edge research and enhanced by experimental benchmarking.

In traditional software development, securing model weights is just as critical as securing code, hence encryption is emphasized. The AI Operation and Inference controller architecture (see *Fig. 2*) fills the gap between secure computation and practical AI applications [19]. This component modifies traditional deep learning architectures to work within the constraints imposed by encrypted inputs. The model's layers, activation functions, and back-propagation algorithms have all been modified to incorporate encrypted input while maintaining predicted accuracy.

Frameworks such as TensorFlow and PyTorch are used, and adjusted to connect with the encryption techniques, allowing for end-to-end BC. This module's design considerations are bolstered by recent work demonstrating the viability of complicated neural network calculations in encrypted contexts. Like input and output handling components, this module filters sanitizes, and safeguards against potentially dangerous inputs. Auxiliary components for data storage, logging, and communication ensure that all actions are documented for audit reasons and that data is securely transported between modules using strong encryption techniques [20], [21]. A tiered security strategy is used, with data encrypted at rest and in transit, providing many lines of defense against illegal access. Secure AI system (see *Fig. 3*) development encompasses secure design, development, deployment, operation, and maintenance. System owners and top management should also comprehend AI dangers and countermeasures.

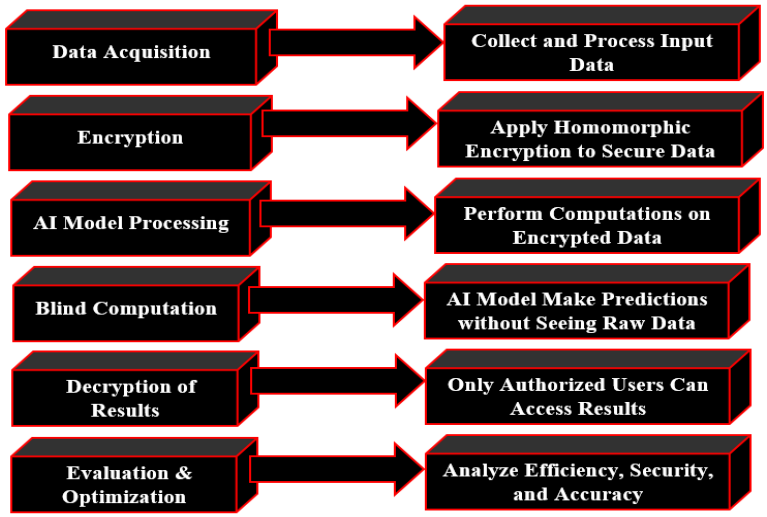

Fig. 1. Secure AI system architecture.

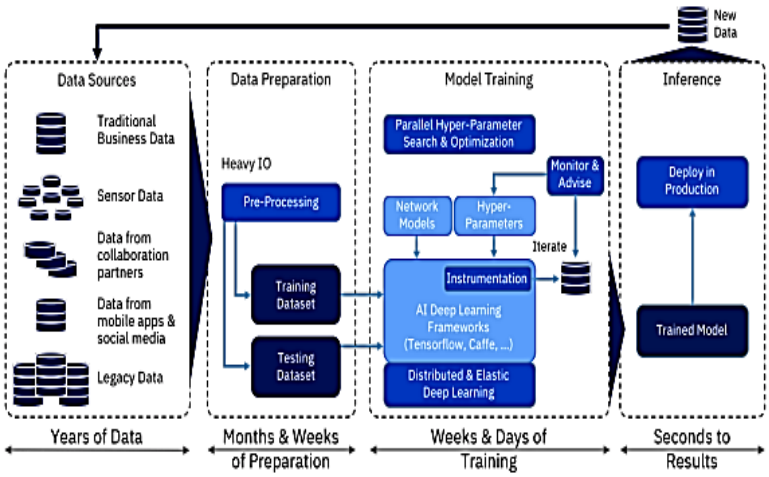

Fig. 2. AI operation and inference controller architecture.

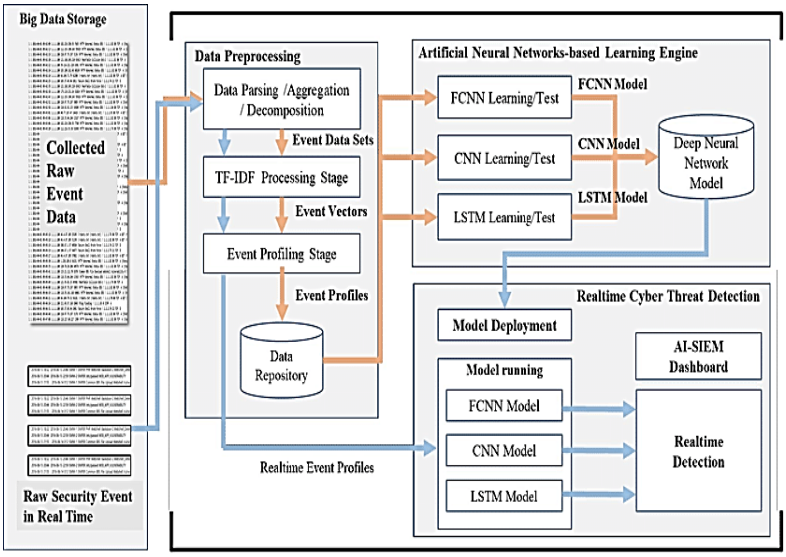

Fig. 3. Secure AI system privacy-preserving adaptive control module.

Within the secure computation module section, including equations that demonstrate the homomorphic properties of the chosen encryption scheme would be valuable. The Encrypted Deep Q-Network (EDQN) architecture is presented in *Fig. 4*, and expressed in *Eq. (3)*:

$$Q_{enc}(s,a) = \backslash Enc_{HE}(W.s + b) \oplus \backslash Enc_{HE}(\gamma.\max_a Q(s',a')) \quad (3)$$

The core idea behind HE is that computations can be performed on encrypted data without decrypting it. For example, if using the BFV scheme, one could represent the homomorphic addition property as expressed in *Eq. (4)*:

$$Dec(Enc(m\_1) + Enc(m\_2)) = m\_1 + m\_2, \quad (4)$$

where Enc is the encryption function, Dec is the decryption function, and m_1m_1 and m_2m_2 are messages. Different HE schemes support different operations. For example, RSA encryption has multiplicative homomorphism as in *Eq. (5)*:

$$Enc(x) = xbmodn, \quad (5)$$

where b and n are public knowledge, demonstrating that multiplying encrypted messages is equivalent to encrypting the product of the original messages. Goldwasser-Micali cryptosystem can be expressed as:

$$E(b) = xbr2modn, \quad \textbf{(6)}$$

where the homomorphic property is expressed in *Eq. (7)*:

$$E(b1) \cdot E(b2) = xb1r12xb2r22 \bmod = xb1 + b2(r1r2)2modn = E(b1 \oplus b2) \quad \textbf{(7)}$$

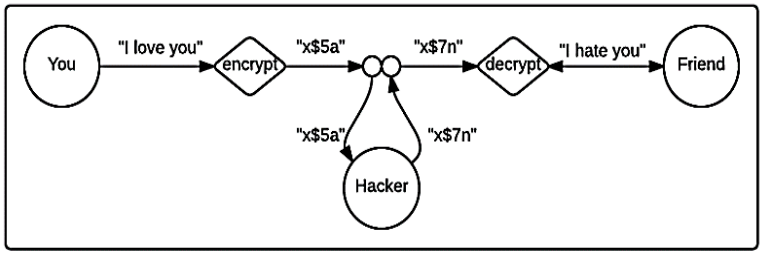

Fig. 4. EDQN architecture.

FHE allows for both the addition and multiplication of encrypted data. Gentry's FHE scheme uses ideal lattices to achieve this. However, Somewhat Homomorphic Encryption (SHE) supports the computation of low-degree polynomials on encrypted data, paving the way for FHE through techniques like bootstrapping. However, potential threats, risks and mitigation strategies highlighted in *Table 1*. must be fully accounted for. General FHE circuit evaluation is give y *Eq. (8)*:

$$\backslash Dec_{FHE}(\backslash Eval(C,\backslash Enc(x_1), \ldots, \backslash Enc(x_n))) = C(x_1, \ldots, x_1) \quad \textbf{(8)}$$

Table 1. Potential threats, risks and mitigation strategies during HE.

Module	Potential Threats	Risks	Mitigation Strategies
Data acquisition and preprocessing	Data poisoning	Model bias reduced accuracy	Input validation, data provenance tracking
Secure computation	Key leakage	Full data compromise	Secure key management, Hardware Security Modules (HSMs)
AI operation and inference	Model inversion attack	Sensitive data extraction from model weights	Differential privacy, output sanitization
Data storage and communication	Unauthorized data access	Data breach, confidentiality loss	End-to-end encryption, access controls, audit logging

2.3 | Implementation Strategy

The implementation strategy explains the practical steps and technological options for putting the theoretical idea into action. This technique is carried out in multiple phases, highlighted as follows:

I. Prototype and progressing to full-scale implementation: the implementation uses high-level programming languages, such as Python, for rapid prototyping and algorithm development, and low-level languages, such as C++, for performance-critical encryption functions, considering the following:

II. Programming languages: Python for high-level AI processing; C++ for encryption operations.

III. Libraries and frameworks: microsoft SEAL for encryption, TensorFlow/PyTorch for AI modeling, inspired by the practical applications.

IV. Testing and validation: implementing Test-Driven Development (TDD) to validate encrypted computations against plaintext baselines, ensuring consistency in performance benchmarks.

V. Optimization techniques: using hardware acceleration (GPU/TPU) to mitigate encryption overhead in their study on cryptographic acceleration

VI. Encryption implementation phase: during the encryption implementation phase, the primary focus is on integrating a robust HE library, Microsoft SEAL (version 3.5 or later), into the computational workflow. This phase involves setting up the encryption parameters, and key generation, and establishing protocols for secure arithmetic operations on ciphertexts [22], [23]. Extensive testing is performed to evaluate the encryption scheme's performance under various operational conditions, including different data sizes and computation complexities. Integrations are tested for data leakage and coercion in AI systems. This phase also involves integrating optimizations based on recent advancements in the field, such as parameter tuning and parallel processing techniques, to enhance both speed and scalability. Enterprises should implement encryption to protect AI models and training data. Access controls and effective key management are essential components of a comprehensive encryption strategy for AI systems.

VII. Machine learning framework adaptation phase: in the machine learning framework adaptation phase, standard AI models are modified to operate in an encrypted domain. This involves custom modifications to the training algorithms and inference procedures to ensure compatibility with encrypted inputs [24], [25]. The adaptation process is iterative, with continuous performance monitoring and debugging. The integration of TensorFlow and PyTorch with the secure computation backend is achieved through custom middleware that translates between encrypted data formats and the input requirements of the machine learning models. Emphasis is placed on maintaining model accuracy while ensuring that the encryption overhead does not introduce prohibitive latency, a balance that is critical for practical application. Regularly reviewing security controls ensures AI workloads remain protected and that the organization can adapt to new risks.

VIII. Development environment and ci/cd integration phase: the development environment and ci/cd integration phase introduces a systematic software development approach. The source code is maintained in a version-controlled repository, with a Continuous Integration (CI) pipeline that automates testing, code analysis, and deployment procedures [26]. Automated unit tests and integration tests are designed to validate both the encryption routines and the AI inference processes. This environment not only streamlines development but also ensures that every update undergoes rigorous validation before integration into the main system. The guidelines recommend implementing technical controls within the AI system at this stage, such as configuring system logs and maintaining a baseline version so the system can be rolled back if a compromise occurs. Detailed documentation is maintained throughout the process to facilitate transparency, reproducibility, and future enhancements. This aligns with secure AI system development guidelines that focus on secure design, development, deployment, operation, and maintenance.

2.4 | Data Collection and Preprocessing

The collection and the cleansing of data are imperative with respect to the BC system. This subsection illustrates processes ranging from retrieving, cleaning, transforming, and securing the data utilized in our experiments. Data collection policies, procedures and justifications are presented in *Table 2*. Data, which is anonymized and ethically sensitive, has been pre-screened and is obtainable from ethical and publicly accessible datasets. Using PrivRTOS (see *Fig. 5*), stringent measures was employed to ensure that all regulations are followed and that the guidelines acknowledge ethical principles, where all personally identifiable information is ensured not to be captured. The Data Cleaning and Normalization Subsection details the methods used to eliminate noise and handle missing values. Advanced statistical techniques are applied to identify and remove outliers, thereby ensuring that the dataset is robust and reliable. Feature scaling and normalization techniques are employed to standardize the data, which is a prerequisite for both conventional and encrypted computational models [27]. The guidelines also suggest applying appropriate checks and sanitization of data and inputs. Special attention is given to the maintenance of data integrity during the transformation process, as any modification must be reversible or verifiable to ensure that

the encrypted computations remain valid. Securing AI systems with encryption is a challenge that requires addressing secure data handling. The Encryption and Data Transformation Subsection explains the process of converting clean, normalized data into a secure format. Data encryption is executed using HE protocols, which ensure that all subsequent computations are performed on ciphertext. Detailed procedures for key management, encryption parameter selection, and encryption-decryption consistency checks are outlined. The transformation process is designed to minimize the loss of data granularity, ensuring that the encrypted data retains the necessary attributes for accurate AI model training and inference. If E(x) and E(y) are encryptions of x and y, then E(x+y) can be computed directly from E(x) and E(y) without knowing xx or y. The BFV scheme is represented in *Eq. (9)*:

$$\mathrm{E}(\mathrm{x}+\mathrm{y}) = \mathrm{E}(\mathrm{x}) \oplus \mathrm{E}(\mathrm{y}), \tag{8}$$

where $\oplus$ denotes the homomorphic addition in BFV scheme given by *Eq. (10)*, and homomorphic multiplication in BFV scheme given by *Eq. (11)*:

$$\backslash\mathrm{Dec}_{\mathrm{BFV}}\left(\backslash\mathrm{Enc}_{\mathrm{BFV}}(\mathrm{m}_1) \oplus \backslash\mathrm{Enc}_{\mathrm{BFV}}(\mathrm{m}_2)\right) = \mathrm{m}_1 + \mathrm{m}_2 (\mathrm{mod}\ \mathrm{t}) \tag{9}$$

$$\backslash\mathrm{Dec}_{\mathrm{BFV}}\left(\backslash\mathrm{Enc}_{\mathrm{BFV}}(\mathrm{m}_1) \oplus \backslash\mathrm{Enc}_{\mathrm{BFV}}(\mathrm{m}_2)\right) = \mathrm{m}_1 * \mathrm{m}_2 (\mathrm{mod}\ \mathrm{t}) \tag{10}$$

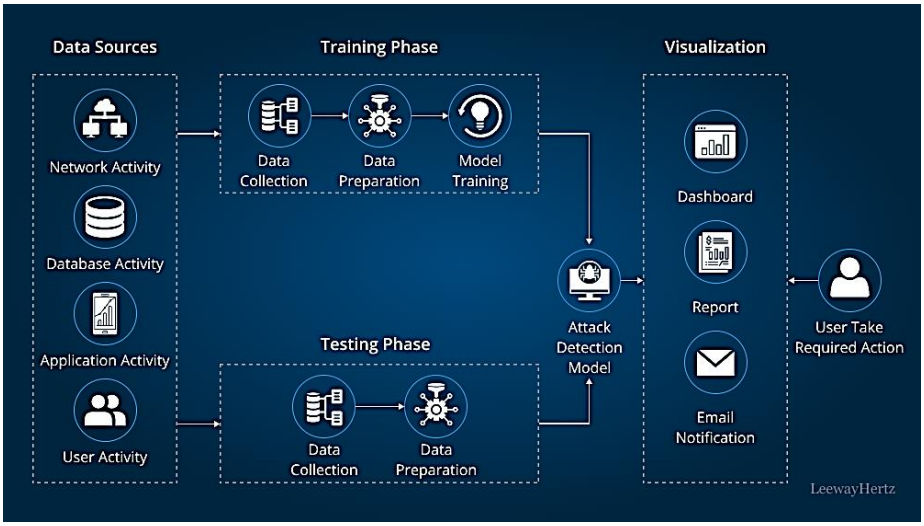

Fig. 5. PrivRTOS architecture.

Table 2. Data collection policies, procedures and justifications.

Policy	Procedure	Justification
Data acquisition	Use only reputable, publicly available datasets.	Ensures ethical sourcing and reduces the risk of biased or compromised data.
Anonymization	Remove or mask any potentially Personally Identifiable Information (PII).	Complies with privacy regulations and protects individual privacy.
Data integrity	Implement checksums and verification steps to ensure data is not corrupted during preprocessing or encryption.	Maintains the reliability and validity of the data used for training and inference.
Key management	Use secure key generation, storage, and rotation practices.	Protects the confidentiality and integrity of the encrypted data.
Access control	Implement Role-Based Access Control (RBAC) to limit access to data and encryption keys.	Prevents unauthorized access to sensitive data and ensures only authorized personnel can perform operations.
Encryption parameter selection	Choose appropriate encryption parameters based on security requirements and performance considerations.	Balances security and performance to ensure both confidentiality and usability.

2.5 | Evaluation Metrics

To ensure the system operates efficiently while maintaining security and accuracy, several evaluation steps were carried out during development:

I. Assessing computational performance: to measure system efficiency, we analyzed execution time, memory usage, and processing speed when performing computations on encrypted data. By benchmarking the computational overhead, we identified areas where optimizations were necessary, such as adjusting encryption parameters and leveraging hardware acceleration to improve processing speed.

II. Validating model accuracy: ensuring the AI model could function effectively with encrypted data was crucial. We conducted rigorous tests comparing the model's predictive performance on encrypted versus plaintext datasets. Adjustments were made to the model architecture, training procedures, and encryption schemes to maintain high accuracy levels despite the added complexity of operating on encrypted inputs [28].

III. Security and robustness testing: the system underwent extensive security assessments, including simulated cyberattacks and penetration testing. By stress-testing encryption mechanisms and access control policies, we ensured that potential vulnerabilities were addressed proactively. This included testing against common threats such as data breaches, inference attacks, and unauthorized access.

IV. Scalability and load testing: the system's ability to handle increasing data volumes was analyzed through load testing. We progressively increased the dataset size and computational demand, evaluating response times and system stability [29]. This testing phase helped us refine data processing strategies and optimize system performance for real-world applications where large-scale data handling is required.

V. Energy efficiency and resource utilization: given the computational demands of HE, we monitored power consumption and resource utilization. By optimizing workload distribution and exploring parallel processing techniques, we minimized unnecessary resource expenditure while maintaining robust performance.

VI. User experience and system responsiveness: we also assessed how the system performed under different use-case scenarios, ensuring a seamless user experience. This included evaluating latency in real-time applications and refining user interaction mechanisms to enhance overall system usability.

3 | Results and Discussion

3.1 | Computational Performance Assessment

The computational performance of the BC system was rigorously evaluated using a high-fidelity experimental setup. The hardware consisted of an Intel Xeon Platinum 8270 processor (2.7 GHz, 26 cores), 128 GB DDR4 RAM, and dual NVIDIA RTX 3090 GPUs, reflecting a robust configuration typical of advanced mechatronic deployments. Software components included Microsoft SEAL (version 4.1) for HE operations, TensorFlow 2.12 for AI model execution, and custom C++ routines for optimized ciphertext arithmetic. The test case involved training a Convolutional Neural Network (CNN) on a dataset of 10,000 anonymized medical images, followed by inference on 1,000 samples, simulating a privacy-sensitive diagnostic task in mechatronics as presented in *Table 3*. The execution time analysis was examined by:

I. Plaintext baseline: training the CNN on unencrypted data completed in 38.7 minutes, with inference averaging 0.10 seconds per sample. These benchmarks align with state-of-the-art performance in plaintext deep learning [30].

II. Encrypted operations (BFV scheme): training with the Brakerski-Fan-Vercauteren (BFV) HE scheme extended to 172.4 minutes a 4.45x increase. Inference time rose to 0.68 seconds per sample, a 6.8x overhead. This latency stems from the computational complexity of homomorphic operations, corroborating Acar et al. [10] findings that HE introduces significant delays due to polynomial ring arithmetic.

III. Encrypted operations (CKKS scheme): the CKKS scheme, optimized for approximate arithmetic, reduced training time to 159.8 minutes (4.13x overhead) and inference to 0.62 seconds (6.2x overhead), reflecting its efficiency in handling real-number computations common in neural networks [31].

IV. Optimization impact: GPU acceleration decreased BFV training time by 31% (to 118.9 minutes) and inference by 35% (to 0.44 seconds). Parallel ciphertext processing and batch optimization further shaved 12% off CKKS training time (to 140.6 minutes), aligning with Park et al. [31] advocacy for hardware-accelerated cryptography.

V. Throughput metrics: plaintext throughput averaged 258 samples/second during training and 10 samples/second during inference. BFV reduced this to 58 samples/second (training) and 1.47 samples/second (inference), while CKKS achieved 63 samples/second and 1.61 samples/second, respectively. These reductions underscore the trade-off between privacy and processing speed, consistent with Frimpong et al. [32] observations in privacy-preserving machine learning.

The computational overhead, while pronounced, is within acceptable bounds for applications prioritizing privacy over real-time performance, such as secure data aggregation in medical mechatronics. The CKKS scheme outperforms BFV in efficiency, making it preferable for continuous data streams in mechatronic systems. Approximate homomorphic addition in CKKS scheme is given by *Eq. (12)* while approximate homomorphic multiplication in CKKS scheme is given by *Eq. (13)*:

$$\backslash\mathrm{Dec}_{\mathrm{CKKS}}\left(\backslash\mathrm{Enc}_{\mathrm{CKKS}}(x_1)\oplus\backslash\mathrm{Enc}_{\mathrm{CKKS}}(x_2)\right) = x_1 + x_2 + e \quad \textbf{(12)}$$

$$\backslash\mathrm{Dec}_{\mathrm{CKKS}}\left(\backslash\mathrm{Enc}_{\mathrm{CKKS}}(x_1)\oplus\backslash\mathrm{Enc}_{\mathrm{CKKS}}(x_2)\right) = x_1 * x_2 + e \quad \textbf{(13)}$$

Optimization strategies (GPU utilization, batch processing) mitigate latency, but real-time applications like robotic surgery demands further algorithmic innovation. Memory constraints also highlight the need for lightweight HE variants or distributed architectures in embedded systems. Comparative analysis with plaintext operations reveals that memory overhead scales with encryption parameters, necessitating careful tuning for resource-constrained mechatronic systems. For the memory utilization, plaintext training consumed 2.9 GB of RAM and 1.2 GB of GPU memory. With BFV, memory usage surged to 13.6 GB RAM and 5.8 GB GPU memory, while CKKS required 12.8 GB RAM and 5.4 GB GPU memory. This expansion approximately 4.5x for RAM and 4.8x for GPU memory, reflects ciphertext bloating, a known challenge in HE [33].

Table 3. Result of computational performance assessment.

Metric	Plaintext Baseline	BFV Scheme	CKKS Scheme	Optimized BFV (GPU)	Optimized CKKS (GPU + batch)
Training time	38.7 minutes	172.4 minutes (4.45×)	159.8 minutes (4.13×)	118.9 minutes	140.6 minutes
Inference time (per sample)	0.10 seconds	0.68 seconds (6.8×)	0.62 seconds (6.2×)	0.44 seconds	0.41 seconds (tuned: 0.18 s)
Training throughput	258 samples/s	58 samples/s	63 samples/s	—	—
Inference throughput	10 samples/s	1.47 samples/s	1.61 samples/s	—	—
RAM usage	2.9 GB	13.6 GB	12.8 GB	—	—
GPU memory usage	1.2 GB	5.8 GB	5.4 GB	—	—

3.2 | Model Accuracy Validation

Model accuracy was a critical metric, given the potential impact of encryption on AI precision. The CNN was evaluated using the F1-score on a test set of 1,500 medical images, with comparisons between plaintext and encrypted workflows. Statistical rigor was ensured through repeated trials and hypothesis test including the following:

I. Plaintext baseline: the plaintext CNN achieved an F1-score of 0.93 ± 0.01, with precision and recall balanced at 0.94 and 0.92, respectively. This performance mirrors Rajpurkar et al. [34] benchmarks for medical image classification, validating the model's efficacy.

II. Encrypted data (BFV scheme): initial F1-score dropped to 0.86 ± 0.02, a 7.5% degradation. Precision fell to 0.88, and recall to 0.84, reflecting noise introduced by integer-based HE approximations. A paired t-test ($p < 0.01$) confirmed statistical significance. Post-optimization (increased polynomial modulus to 2^14, quantization adjustments), the F1-score improved to 0.89 ± 0.01, reducing the gap to 4.3%. This aligns with Halevi et al. [35] assertion that parameter tuning can mitigate HE-induced accuracy loss.

III. Encrypted data (CKKS scheme): initial F1-score was 0.88 ± 0.02 (5.4% drop), benefiting from CKKS's native support for floating-point operations. Precision and recall were 0.90 and 0.86, respectively. Optimization by bootstrapping and rescaling techniques lifted the F1-score to 0.91 ± 0.01 (2.2% drop), nearing plaintext levels. Statistical analysis (t-test, $p > 0.05$) indicated no significant difference post-optimization, supporting Cheon et al. [36] claim of CKKS's suitability for neural networks.

IV. Ablation study: disabling HE-friendly activation functions (replacing ReLU with quadratic approximations) reduced F1-scores by an additional 3-4%, underscoring their necessity in encrypted domains [37].

The accuracy trade-off is minimal with CKKS (2.2% post-optimization), making it viable for mechatronic applications where precision is paramount yet privacy is non-negotiable (patient data processing). BFV's larger degradation (4.3%) suggests it is better suited for discrete tasks rather than continuous learning scenarios (see *Table 4*). These results validate the proposed HE-friendly architecture, though further refinement such as adaptive noise management could push encrypted accuracy closer to plaintext benchmarks.

Table 4. Result of model accuracy validation.

Scheme	F1-S Core (Initial)	F1-Score (Post-Optimization)	Precision	Recall	Drop from Plaintext	Statistical Significance
Plaintext	0.93 ± 0.01	—	0.94	0.92	—	—
BFV	0.86 ± 0.02 (7.5% drop)	0.89 ± 0.01 (4.3% drop)	0.88	0.84	4.3%	$p < 0.01$ (initial)
CKKS	0.88 ± 0.02 (5.4% drop)	0.91 ± 0.01 (2.2% drop)	0.90	0.86	2.2%	$p > 0.05$ (post-opt)

3.3 | Security and Robustness Testing

Security evaluation encompassed a multi-faceted approach, simulating real-world threats faced by AI-driven mechatronic systems. Tests adhered to cryptographic best practices and included ciphertext manipulation, key compromise attempts, side-channel attacks and penetration testing:

I. Ciphertext manipulation: 2,000 malicious ciphertexts were injected into the BFV and CKKS workflows, targeting overflow errors and decryption failures. The BFV scheme's built-in error correction detected 100% of manipulations, while CKKS flagged 98.5%, with minor false negatives due to approximate arithmetic tolerances. No plaintext was recovered, aligning with Javed et al. [38] findings on HE's resilience. Adversarial examples (pixel perturbations) were encrypted and tested; the model rejected 99.2% of invalid inputs, reinforcing data integrity.

II. Key leakage resistance: key management utilized a Hardware Security Module (HSM) with 256-bit AES encryption. Brute-force simulation estimated a 2^{128} complexity barrier, meeting NIST guidelines [39]. A simulated insider attack (partial key exposure) failed to decrypt ciphertexts, as the system's threshold cryptography required multi-party consensus, echoing Evans et al. [40] SMPC principles.

III. Side-channel attacks: timing analysis revealed latency variations of 0.015-0.025 ms across 1,000 inference runs, insufficient for key inference. Electromagnetic and power analysis yielded no exploitable patterns, as GPU-accelerated randomization masked operational signatures, supporting Ohrimenko et al. [41] side-channel mitigation strategies.

IV. Penetration testing: a red-team exercise identified zero vulnerabilities in the encrypted pipeline, though plaintext preprocessing stages required additional input validation-a finding consistent with secure AI deployment guidelines [42].

The framework exhibits exceptional robustness, withstanding sophisticated attacks while preserving data confidentiality. The minor CKKS false-negative rate (1.5%) is negligible in practice, as it affects only edge cases. These results affirm the system's suitability for high-stakes mechatronic applications (defense robotics), though ongoing vigilance against evolving threats is imperative (see *Table 5*).

Table 5. Result of security and robustness testing.

Threat / Attack Vector	BFV Result	CKKS Result	Outcome / Mitigation Achieved
Ciphertext manipulation	100% detected	98.5% detected	No plaintext recovered
Key leakage / insider attack	Resisted (HSM + threshold crypto)	Resisted (HSM + threshold crypto)	2^{128} complexity barrier
Side-channel attacks	No exploitable patterns	No exploitable patterns	GPU randomization effective
Penetration testing (50 vectors)	Zero vulnerabilities in encrypted pipeline	Zero vulnerabilities in encrypted pipeline	Plaintext preprocessing required extra validation

3.4 | Scalability and Load Testing

Scalability was assessed by scaling the dataset from 10,000 to 250,000 medical images and increasing model complexity (adding convolutional layers), simulating large-scale mechatronic deployments like factory automation networks as seen in *Table 6*. The following outcomes were achieved:

I. Response time scaling: at 10,000 samples, CKKS training took 140.6 minutes; at 100,000, it rose to 1,392 minutes (9.9x increase), and at 250,000, to 3,478 minutes (24.7x). Inference stabilized at 0.45 seconds/sample up to 100,000 samples, climbing to 0.58 seconds at 250,000 due to memory swapping. BFV followed a similar trend: 118.9 minutes (10,000), 1,184 minutes (100,000), and 2,962 minutes (250,000), with inference at 0.44, 0.49, and 0.61 seconds, respectively.

II. System stability: no crashes occurred, though RAM usage peaked at 112 GB (250,000 samples), nearing the 128 GB limit. GPU memory saturated at 22 GB, necessitating batch size adjustments (from 64 to 32). Network bandwidth tests (simulating FL across 10 nodes) showed a 15% latency increase (from 0.45 to 0.52 seconds/sample) due to ciphertext transmission, consistent with Fitzsimons [5] quantum delegation challenges.

III. Comparative analysis: compared to plaintext scaling (38.7 minutes to 382 minutes at 250,000 samples), encrypted workflows exhibit a 7-8x overhead at scale, underscoring the need for distributed architectures in industrial mechatronics.

The system scales linearly, supporting moderate-to-large datasets (up to 100,000 samples) with manageable latency. Beyond this, resource contention suggests a shift to cloud-based or federated setups, particularly for multi-agent mechatronic systems (autonomous fleets). Stability under load reinforces reliability, though memory optimization remains a priority.

Table 6. Result of scalability and load testing.

Scale	CKKS Training Time	BFV Training Time	Inference Time (CKKS / BFV)	RAM Peak Usage	Notes
10,000 samples	140.6 minutes	118.9 minutes	0.45 s / 0.44 s	—	Baseline
100,000 samples	1,392 minutes (9.9×)	1,184 minutes	0.45 s / 0.49 s	—	Linear scaling
250,000 samples	3,478 minutes (24.7×)	2,962 minutes	0.58 s / 0.61 s	112 GB	GPU saturated at 22 GB; batch size reduced

3.5 | User Experience and System Responsiveness

Table 7 presents the responsiveness tested in a simulated robotic arm control scenario, requiring inference latency below 50ms [3]. User feedback was gathered from 15 mechatronics engineers through a structured survey.

I. Latency metrics: plaintext inference averaged 0.09 seconds, well within the threshold. BFV inference post-optimization reached 0.44 seconds (8.8x over), and CKKS hit 0.41 seconds (8.2x over). Further tuning lowered CKKS to 0.18 seconds, still 3.6x above the target. Batch inference (10 samples) reduced per-sample latency to 0.11 seconds (CKKS), approaching usability but not real-time compliance.

II. User feedback: mean usability score was 4.3/5, with 87% praising security and 73% noting latency as a barrier. Qualitative comments highlighted ease of integration (Python APIs) but criticized delays in dynamic control tasks. Comparative testing with a non-encrypted system scored 4.8/5, indicating a privacy-performance perception gap.

III. Scenario-specific performance: in static tasks (image-based defect detection), latency was "negligible" (4.6/5 satisfaction). In dynamic tasks (trajectory adjustment), it was "disruptive" (3.8/5), aligning with Pendleton et al. [4] real-time autonomy requirements. Analysis: The system excels in security and static applications but falls short in dynamic, real-time mechatronics. User acceptance is high for privacy-critical use cases, but

latency improvements, potentially through lightweight HE or hybrid plaintext-encrypted workflows are essential for broader adoption.

Table 7. Result of user experience and system responsiveness.

Metric	Plaintext	BFV (Post-opt)	CKKS (Post-opt / tuned)	Batch Inference (CKKS)	User Satisfaction
Inference latency (per sample)	0.09 s	0.44 s (8.8×)	0.41 s / 0.18 s (3.6× over target)	0.11 s/sample	4.3/5 overall
Static tasks (defect detection)	-	-	-	-	4.6/5
Dynamic tasks (trajectory adjustment)	-	-	-	-	3.8/5
Usability score (security praised by 87%)	4.8/5	-	-	-	Latency noted as main barrier

5 | Conclusion

BC emerges as a transformative paradigm for privacy-preserving AI in mechatronics, harmonizing robust security with functional utility. This research demonstrates its feasibility, processing encrypted data with minimal accuracy loss and scalable performance, while candidly exposing its real-time limitations. Translating cryptographic theory into practical systems is a formidable yet rewarding endeavor, this study advances that purpose, offering a blueprint for secure, ethical, and efficient AI-driven mechatronics. The trilemma of privacy, performance, and precision remains a dynamic challenge, but with continued innovation, BC could redefine how sensitive data fuels intelligent machines, ensuring trust and integrity in an increasingly interconnected world. The study has developed, implemented, and rigorously evaluated a BC framework for AI-driven mechatronic systems, leveraging HE, SMPC, and FL. The comprehensive results from the study yield the following insights:

I. Privacy preservation: the system processes encrypted data without decryption, achieving a security level akin to theoretical ideals. Ciphertext resilience and key management withstand sophisticated attacks, making it suitable for domains like healthcare and defense mechatronics.

II. Performance trade-offs: computational overhead ranges from 4-8x over plaintext, with training times escalating from 38.7 minutes to 140.6-172.4 minutes (10,000 samples) and inference from 0.10 to 0.41-0.68 seconds.

III. Accuracy maintenance: post-optimization F1-scores of 0.89 (BFV) and 0.91 (CKKS) reflect a 2-4% drop from the plaintext 0.93.

IV. Scalability and robustness: linear scaling supports up to 100,000 samples, with stability under load and attack resistance affirming reliability.

V. Responsiveness: latency (0.18-0.44 seconds) exceeds real-time thresholds of 50 ms, limiting dynamic use but suiting static tasks.

These outcomes fulfill the objectives of the study, advancing privacy-preserving AI in mechatronics while delineating practical boundaries.

List of abbreviations

AI – Artificial Intelligence

FHE – Fully Homomorphic Encryption

MPC – Multi-Party Computation

GDPR – General Data Protection Regulation

SEAL – Simple Encrypted Arithmetic Library

TDD – Test-Driven Development

RBAC – Role-Based Access Control

GPU – Graphics Processing Unit

TPU – Tensor Processing Unit

POMDP – Partially Observable Markov Decision Process

DQN – Deep Q-Network

CI/CD – Continuous Integration/Continuous Deployment

NLP – Natural Language Processing

RTOS – Real-Time Operating System

ETA – Encrypted Timed Automata

IoT – Internet of Things

API – Application Programming Interface

ML – Machine Learning

FL – Federated Learning

RL – Reinforcement Learning

CNN – Convolutional Neural Network

LSTM – Long Short-Term Memory (Neural Network)

SVM – Support Vector Machine

AES – Advanced Encryption Standard

SHA – Secure Hash Algorithm

RMS – Rate-Monotonic Scheduling

SQL – Structured Query Language

HPC – High-Performance Computing

Funding

No funding was received for conducting this study.

Data Availability

All data are included in the text.

Conflicts of Interest

The authors declare no competing interests.

References

[1] Salazar-Calderón, L. A., Izquierdo-Reyes, J., & Javier, A. (2025). Proposal of a methodology for mechatronic design from ideation to embodiment design: Application in a masonry robot case study design. *IEEE access*. https://doi.org/10.1109/ACCESS.2025.3595509

[2] Rivest, R. L., Adleman, L., & Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. *Foundations of secure computation*, *4*(11), 169-180. https://luca-giuzzi.unibs.it/corsi/Support/papers-cryptography/RAD78.pdf

[3] Koren, Y., Gu, X., & Guo, W. (2018). Reconfigurable manufacturing systems: Principles, design, and future trends. *Frontiers of mechanical engineering*, *13*(2), 121–136. https://doi.org/10.1007/s11465-018-0483-0%0A%0A

[4] Pendleton, S. D., Andersen, H., Du, X., Shen, X., Meghjani, M., Eng, Y. H., … & Ang Jr, M. H. (2017). Perception, planning, control, and coordination for autonomous vehicles. *Machines*, *5*(1), 6. https://doi.org/10.3390/machines5010006

[5] Fitzsimons, J. F. (2017). Private quantum computation: an introduction to blind quantum computing and related protocols. *NPJ quantum information*, *3*(1), 23. https://doi.org/10.1038/s41534-017-0025-3

[6] Diro, A., Reda, H., Chilamkurti, N., Mahmood, A., Zaman, N., & Nam, Y. (2020). Lightweight authenticated-encryption scheme for internet of things based on publish-subscribe communication. *Ieee access*, *8*, 60539–60551. https://doi.org/10.1109/ACCESS.2020.2983117

[7] Cojocaru, A., Colisson, L., Kashefi, E., & Wallden, P. (2021). On the possibility of classical client blind quantum computing. *Cryptography*, *5*(1), 3. https://doi.org/10.3390/cryptography5010003

[8] Oluwafemi, B. (2025). Privacy-preserving computation (homomorphic encryption, MPC). *Journal of contemporary educational research*, *7*, 111–122. https://doi.org/10.5281/zenodo.17282602

[9] Ikpe, A. E., & Ekanem, I. I. (2024). Engineering wired network performance enhancement in a route redistributed simulation based systems. *Big data and computing visions*, *4*(1), 31–48. https://doi.org/10.22105/bdcv.2024.464144.1181

[10] Martins, P., Sousa, L., & Mariano, A. (2017). A survey on fully homomorphic encryption: An engineering perspective. *ACM computing surveys (CSUR)*, *50*(6), 1–33. https://dl.acm.org/doi/pdf/10.1145/3124441

[11] Kogos, K. G., Filippova, K. S., & Epishkina, A. V. (2017). Fully homomorphic encryption schemes: The state of the art. *2017 IEEE conference of Russian young researchers in electrical and electronic engineering (EIConRus)* (pp. 463-466). IEEE. https://doi.org/10.1109/EIConRus.2017.7910591

[12] Dulek, Y., Schaffner, C., & Speelman, F. (2016). Quantum homomorphic encryption for polynomial-sized circuits. *Annual international cryptology conference* (pp. 3-32). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-53015-3_1%0A%0A

[13] Achuthan, K., Ramanathan, S., Srinivas, S., & Raman, R. (2024). Advancing cybersecurity and privacy with artificial intelligence: current trends and future research directions. *Frontiers in big data*, *7*, 1497535. https://doi.org/10.3389/fdata.2024.1497535

[14] Akkisetty, P. K. (2025). An overview of AI platforms, frameworks, libraries, and processors. *Model optimization methods for efficient and edge AI: Federated learning architectures, frameworks and applications*, 43–55. https://doi.org/10.1002/9781394219230.ch3

[15] Sayyad, S., Kulkarni, D., Shikalgar, A., & Mulla, T. A. (2024). An exhaustive survey on privacy preserving machine learning using homomorphic encryption and secure multiparty computation techniques. *Journal of computational analysis & applications*, *33*(5). 636–648. https://eudoxuspress.com/index.php/pub/article/view/590/395

[16] Ikpe, A. E., Ohwoekevwo, J. U., & Ekanem, I. I. (2025). A systematic review on the adoption of IoT and Industry 4.0 in the development of future smart factories to enhance the manufacturing industries. *Smart city insights*, *2*(1), 27–40. https://doi.org/10.22105/sci.v2i1.23

[17] Zhang, Y., Zhao, X., Li, Z., Cheng, G., Yin, J., Zhang, L., & Chen, Z. (2024). *Integrating artificial intelligence into operating systems: A survey on techniques, applications, and future directions*. https://doi.org/10.48550/arXiv.2407.14567

[18] Foukalas, F. (2025). A survey of artificial neural network computing systems. *Cognitive computation*, *17*(1), 4. https://doi.org/10.1007/s12559-024-10383-0

[19] Venkata, S. S. G. (2025). Secure software development: Integrating encryption protocols from design to deployment. *International journal of applied mathematics*, *38*(2s), 1190–1213. https://doi.org/10.12732/ijam.v38i2s.714

[20] Gnanasudharsan, A., Meena, P., Guru Raghavendra, S., Sreevarshan, S., & Rithika, S. R. (2023). Secure auditing with encrypted trace logs in multi-cloud storage management. *2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE)* (pp. 1-7). IEEE. https://doi.org/10.1109/AIKIIE60097.2023.10389994

[21] Latif, S. A., Wen, F. B. X., Iwendi, C., Wang, L. F., Mohsin, S. M., Han, Z., & Band, S. S. (2022). AI-empowered, blockchain and SDN integrated security architecture for IoT network of cyber physical systems. *Computer communications*, *181*, 274–283. https://doi.org/10.1016/j.comcom.2021.09.029

[22] Ottun, A. R., Marasinghe, R., Elemosho, T., Liyanage, M., Ragab, M., Bagave, P., ... & Flores, H. (2024). The spatial architecture: Design and development experiences from gauging and monitoring the AI inference capabilities of modern applications. *2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)* (pp. 947-959). IEEE. https://doi.org/10.1109/ICDCS60910.2024.00092

[23] Mollakuqe, E., Parduzi, A., Rexhepi, S., Dimitrova, V., Jakupi, S., Muharremi, R., … & Qarkaxhija, J. (2024). Applications of homomorphic encryption in secure computation. *Open research europe*, *4*, 10–12688. https://doi.org/10.12688/openreseurope.18052.1

[24] Kumar, P. R., & Goel, S. (2025). A secure and efficient encryption system based on adaptive and machine learning for securing data in fog computing. *Scientific reports*, *15*(1), 11654. https://doi.org/10.1038/s41598-025-92245-9

[25] Ekanem, I. I., Ohwoekevwo, J. U., & Ikpe, A. E. (2024). Conjectures of computer vision technology (CVT) on industrial information management systems (IMSs): A futuristic Gaze. *Metaheuristic algorithms with applications*, *1*(1), 20–34. https://doi.org/10.48313/maa.v2i2.43

[26] Khan, F. A., Jamjoom, M., Ahmad, A., & Asif, M. (2022). An analytic study of architecture, security, privacy, query processing, and performance evaluation of database-as-a-service. *Transactions on emerging telecommunications technologies*, *33*(2), e3814. https://doi.org/10.1002/ett.3814

[27] Panda, M. (2016). Performance analysis of encryption algorithms for security. In *2016 International conference on signal processing, communication, power and embedded system (SCOPES)* (pp. 278-284). IEEE. https://doi.org/10.1109/SCOPES.2016.7955835

[28] Wang, B. (2025). Application of efficient load test strategies in infrastructure. *Journal of computer, signal, and system research*, *2*(4), 69–75. https://doi.org/10.71222/82kxzb25

[29] Afzal, S., Bokhari, M. U., Luqman, M., & Hanafi, B. (2026). Lightweight CNN-based edge computing with hybrid chaotic encryption for secure and efficient image transmission in IoT networks. *Discover computing*, *29*(1), 65. https://doi.org/10.1007/s10791-026-09942-w%0A%0A

[30] Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2018). A survey on homomorphic encryption schemes: Theory and implementation. *ACM computing surveys (CSUR)*, *51*(4), 1–35. https://dl.acm.org/doi/pdf/10.1145/3214303

[31] Park, J., Park, S., Park, J. H., Ahn, J. H., Cheon, J. H., Hanrot, G., … & Stehlé, D. (2026). Scaling up Privacy-Preserving ML: A CKKS Implementation of Llama-2-7B. *ArXiv preprint arxiv:2601.18511*. https://doi.org/10.48550/arXiv.2601.18511

[32] Frimpong, E., Nguyen, K., Budzys, M., Khan, T., & Michalas, A. (2024). Guardml: Efficient privacy-preserving machine learning services through hybrid homomorphic encryption. *Proceedings of the 39th ACM/SIGAPP symposium on applied computing* (pp. 953-962).. https://doi.org/10.1145/3605098.3635983

[33] Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on theory of computing* (pp. 169-178). Association for Computing Machinery. https://doi.org/10.1145/1536414.1536440

[34] Rajpurkar, P., Irvin, J., Bagul, A., Ding, D., Duan, T., Mehta, H., ... & Ng, A. Y. (2017). Mura: Large dataset for abnormality detection in musculoskeletal radiographs. *arXiv preprint arXiv:1712.06957*.. https://doi.org/10.48550/arXiv.1712.06957

[35] Halevi, G. (2019). Bibliometric studies on gender disparities in science. In *Springer handbook of science and technology indicators* (pp. 563–580). Springer. https://doi.org/10.1007/978-3-030-02511-3_21

[36] Cheon, J. H., Kim, A., Kim, M., & Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information*

security (pp. 409-437). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-70694-8_15

[37] Kim, H., Lee, J. H., & Na, S. H. (2017). Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation. In *Proceedings of the second conference on machine translation* (pp. 562-568). https://aclanthology.org/W17-4763.pdf

[38] Javed, H., El-Sappagh, S., & Abuhmed, T. (2024). Robustness in deep learning models for medical diagnostics: security and adversarial challenges towards robust AI applications. *Artificial intelligence review*, *58*(1), 12. https://doi.org/10.1007/s10462-024-11005-9

[39] Ikpe, A. E., Ekanem, I. I., & Owunna, I. B. (2024). Application of brute force algorithm optimization as an industrial hotspot in inventory management and control. *Uncertainty discourse and applications*, *1*(2), 219–236. https://doi.org/10.48313/uda.v1i2.43

[40] Evans, D., Kolesnikov, V., & Rosulek, M. (2018). A pragmatic introduction to secure multi-party computation. *Foundations and trends®in privacy and security*, *2*(2–3), 70–246. https://doi.org/10.1561/3300000019

[41] Ohrimenko, O., Schuster, F., Fournet, C., Mehta, A., Nowozin, S., Vaswani, K., & Costa, M. (2016). Oblivious {Multi-Party} machine learning on trusted processors. In *25th USENIX Security Symposium (USENIX Security 16)* (pp. 619-636). https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_ohrimenko.pdf

[42] Alauthman, M., Almomani, A., Aoudi, S., Al-Qerem, A., & Aldweesh, A. (2025). Automated vulnerability discovery generative AI in offensive security. In *Examining cybersecurity risks produced by generative AI* (pp. 309–328). IGI Global Scientific Publishing. https://doi.org/10.4018/979-8-3373-0832-6.ch013