Indonesian Journal of Electrical Engineering and Computer Science
Vol. 44, No. 1, October 2026, pp. 63~78
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v44.i1.pp63-78

Threshold homomorphic encryption in multi-tenant cloud environments: provable-prime paillier with finite-field pedersen verifiable secret sharing

Michael Ayitey Junior[1], Peter Appiahene[1], Obed Appiah[2], Kwabena Adu[1]
[1]Department of Information Technology and Decision Sciences, School of Sciences, University of Energy and Natural Resources (UENR), Sunyani, Ghana
[2]Department of Computer Science and Informatics School of Sciences, University of Energy and Natural Resources (UENR), Sunyani, Ghana

Article Info

Article history:

Received Apr 15, 2026
Revised Aug 20, 2026
Accepted Sep 10, 2026

Keywords:

Container isolation
Homomorphic encryption
Multi-tenant cloud
Provable prime generation
Threshold cryptography
Verifiable secret sharing

ABSTRACT

Multi-tenant cloud platforms place mutually distrusting tenants on shared infrastructure, where a stored decryption key is a single point of total compromise. This paper presents a threshold Paillier framework treating parameter correctness, key custody, and execution isolation as one problem. Both factors of the modulus are produced by Maurer's recursive construction and certified by Pocklington's theorem, so each prime of at least 3,072 bits carries a primality certificate and no probabilistic test enters key generation. The Carmichael exponent is never distributed; a derived decryption exponent is shared under finite-field Pedersen verifiable secret sharing over a prime-order subgroup whose order satisfies an explicit no-wrap bound, so every node verifies its share against published commitments and an inconsistent dealer is disqualified. Decryption is a three-of-five threshold operation performed entirely in the exponent under integer Lagrange weights cleared by the factorial of the node count, so the private exponent is never reassembled. The framework was deployed as six containers, one coordinator and five participants communicating point-to-point, and evaluated on a fifteen-byte input. Threshold decryption and reconstruction completed correctly, yielding 1,536 bytes of ciphertext, an execution time of 679.41 seconds, a throughput of 0.0215 bytes per second, and a normalised communication cost of 101.4. Participant containers required 37.33 to 44.68 mebibytes of memory, under 0.6 percent of the limit, and 21 to 22 processes, while computational cost concentrated at the coordinator. Certified parameter correctness and distributed trust are therefore attainable at negligible per-tenant cost, with arithmetic throughput as the deliberate trade.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Michael Ayitey Junior
Department of Information Technology and Decision Sciences, School of Sciences
University of Energy and Natural Resources (UENR)
Sunyani, Ghana
Email: Michael.ayitey@uenr.edu.gh

1. INTRODUCTION

Cloud computing has displaced dedicated hardware provisioning with elastic, on-demand allocation over infrastructure occupied concurrently by unrelated organisations. The economic efficiency of the model derives from this consolidation, and so does its principal security liability, since workloads belonging to distinct administrative domains become co-resident on a common physical substrate. The resulting exposure

is empirically demonstrable rather than conjectural. Field-programmable gate arrays deployed as cloud accelerators are typically operated in multi-tenant configuration, and their programmability and runtime reconfigurability render them susceptible to remote correlation power analysis mounted by a co-resident tenant without physical instrumentation; the effectiveness of such an attack against processors executing AES-128 and SM4 has been shown to vary systematically with the runtime clock frequency, a parameter no individual tenant governs [1]. Containerisation compounds the concern. Containers share the kernel of the host and do not virtualise low-level components such as the processor, which renders them lighter and more flexible than virtual machines, whereas virtual machines remain fully autonomous systems that address low-level requirements more precisely [2]. The property that yields the efficiency therefore also reduces the separation between tenants from a hypervisor-mediated boundary to a set of kernel-level controls. A tenant consequently relinquishes custody of its data to infrastructure it neither owns nor observes, under a threat model in which co-resident tenants and the platform operator must both be treated as potentially adversarial.

Encryption at rest and in transit does not help, since conventional pipelines decrypt before computing. Homomorphic encryption removes that step, and Paillier's additive variant has supported privacy-preserving clustering, smart-grid aggregation, and verifiable federated aggregation far more cheaply than fully homomorphic schemes [3]–[6]. All share one asymmetry: the computation is distributed, the decryption capability is not. The private exponent stays a single stored object whose disclosure retroactively exposes every ciphertext under that public key, and in multi-tenant deployment its holder is usually the platform operator itself.

Cryptographic responses to concentrated custody are established in principle. Distributed key generation dispenses with the privileged dealer by having participants jointly derive a key that no single party ever materialises. The approach has been adopted to remove the single private key generator on which purely quantum blockchain constructions otherwise depend, combined there with a dual-layer signature mechanism spanning quantum and lattice-based primitives [7], and it has been optimised for identity-based cryptography under the SM9 standard, where a threshold system operating without a trusted centre addresses the single point of failure inherent in a conventional key generation centre and reduces communication and computational demand by refining the Paillier share transformation protocol [8]. Secret sharing supplies the complementary mechanism. Hierarchical schemes extend Shamir's construction by organising participants into priority levels, each with its own threshold, such that higher-level participants are drawn upon when a level cannot assemble its quorum, and recent work adds share verification to this structure while demonstrating resistance to identified attacks [9]. These primitives are individually mature. What remains unestablished is their composition with a homomorphic decryption capability within a deployed multi-tenant system, where the shared secret is not an arbitrary value but the exponent governing the decryption mapping itself.

A further assumption warrants scrutiny. The security of a Paillier modulus rests entirely upon the primality of its two factors, yet standard practice establishes primality by probabilistic testing, which bounds the probability of accepting a composite and therefore yields a statistical confidence rather than a proof. Research on cryptographic key generation has concentrated principally on the quality of the randomness entering the process; an investigation of commercial smartwatch sensors as an entropy source, for instance, reports Shannon entropy of 0.997 and min-entropy of 0.918 under motion-triggered acquisition, while noting that cryptographic post-processing remains necessary before the output is fit for key material [10]. Adequate entropy is necessary but does not establish correctness, since a well-seeded generator may still emit a composite that survives a bounded number of probabilistic rounds. Where the decryption capability is partitioned across mutually distrusting tenants, every participant inherits the consequences of a defective modulus and none is positioned to detect that defect independently. Assurance of the cryptographic parameters should therefore constitute a verifiable property rather than an asserted probability, particularly because the failure mode is not graceful: a modulus admitting a composite factor becomes susceptible to factorisation, and the intractability assumption underpinning the construction is invalidated without any observable deviation in the behaviour of the encryption or decryption routines.

Four limitations follow from the foregoing analysis and jointly delimit the present work. First, reliance upon probabilistic primality testing leaves the correctness of the cryptographic input parameters unproven. Second, centralised custody of the private decryption exponent exposes the system to insider action and single-point failure. Third, distributed environments generally lack a mechanism by which a participant may validate the integrity of the key fragment it has been issued. Fourth, evaluations of such schemes seldom report security guarantees, execution performance, and infrastructure cost within a single measured framework, leaving the operational price of the protection unquantified.

This paper addresses these limitations through an integrated architecture for multi-tenant cloud deployment. A provable-prime key generation procedure constructs each factor of the Paillier modulus by Maurer's recursive method and certifies it under Pocklington's theorem, producing a certificate in which a known prime divisor of the candidate's predecessor exceeds the square root of the candidate, so that primality

is accompanied by a verifiable witness and no probabilistic test is invoked. A threshold cryptographic framework then removes centralised custody: the Carmichael value is never distributed, a derived decryption exponent is partitioned among participating nodes, and recovery is performed in the exponent under integer Lagrange coefficients cleared by the factorial of the node count, so that the private exponent is never reassembled and an authorised quorum must cooperate before any plaintext is produced. Pedersen verifiable secret sharing over a prime-order quadratic-residue subgroup supplies the absent integrity mechanism, enabling each node to validate its own fragment against published commitments and permitting an inconsistent dealer to be identified and excluded. The framework is instantiated in a containerised architecture in which each participant occupies an isolated container communicating exclusively over explicit network interfaces, supporting reproducibility and tenant separation, and is evaluated for communication cost, time complexity, and space complexity upon that deployment. Figure 1 shows a multi-tenant cloud architecture and its complexity. The resulting design is intended to inform the construction of privacy-preserving cloud services in domains such as federated analytics and smart metering, in which computation must be delegated but custody of the decryption capability must not be. The remainder of this paper is organised as follows. Section 2 presents the methodology. Section 3 presents experimental analysis, results, and discussion, and 4 presents the conclusion.

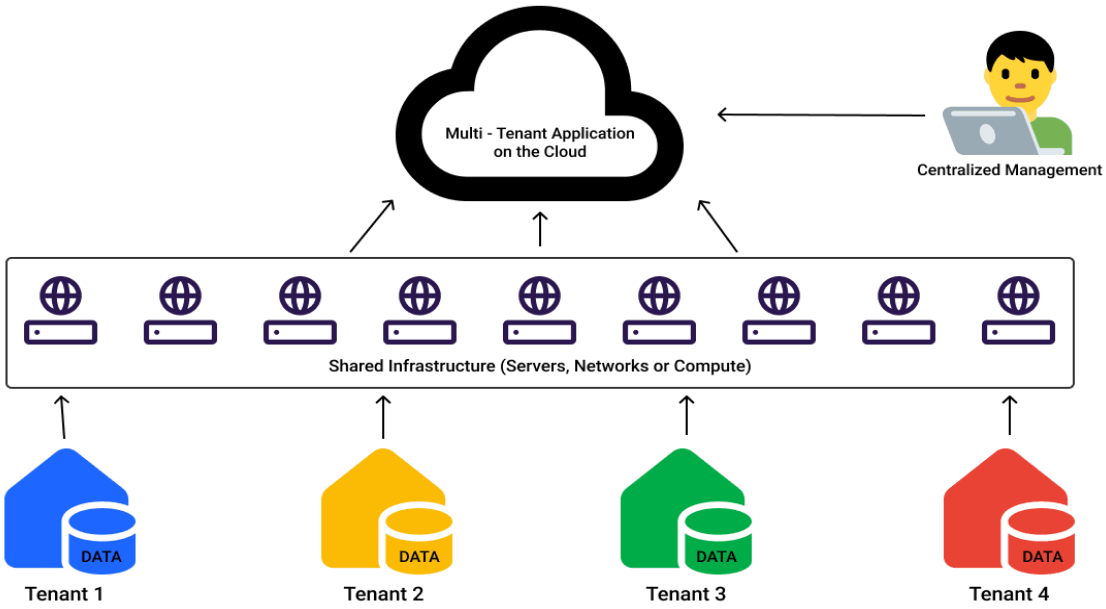


Figure 1. Multitenant architecture and how it works

2. RELATED WORK

Recent work published in IJEEC has addressed the two pillars this paper combines: cryptographic primitive design and lightweight distributed deployment, though not jointly. Pavlović *et al.* [11] construct a hash function over the residue number system, exploiting modular arithmetic across mutually prime moduli to obtain parallelism and high output entropy, and validate it through Hamming weight distribution, bit balance, and avalanche testing. Their design confirms the value of number-theoretic structure in cryptographic construction, but it is a keyless primitive: no key material is generated, distributed, or protected, so parameter certification and key custody do not arise. Malić *et al.* [12] treat the complementary problem, developing a lightweight microservice architecture for small data centre monitoring in which independently deployable services are instrumented for resource observation. That work establishes that fine-grained service decomposition is measurable and operationally viable at small scale, but it carries no cryptographic workload and does not address the confidentiality of tenant data on shared infrastructure.

Threshold constructions have advanced in parallel. Tang *et al.* [13] propose a (t, n)-threshold quantum homomorphic encryption scheme in which k evaluators jointly compute on shared encrypted qubits, demonstrating a (3, 5) instantiation on a quantum cloud platform. The scheme distributes evaluation authority, yet its sharing layer is Shamir secret sharing, which presumes an honest dealer and gives participants no means of detecting an inconsistent share; it further presupposes a quantum network and so offers no classical deployment path. Wang *et al.* [14] address verifiability directly, presenting a changeable-

threshold verifiable secret sharing scheme over the learning with errors problem in which participants confirm share validity and the reconstruction threshold may be altered after distribution. Verifiability is developed for image sharing rather than bound to a homomorphic decryption capability.

Homomorphic encryption in cloud settings has likewise matured. Shylashree and Kumar [15] combine homomorphic threshold detection with privacy-preserving machine learning over wireless sensor data, and [16] propose a federated learning and homomorphic encryption model for multi-tenant cloud platforms, reporting reduced encryption and communication overhead through hierarchical key mapping and dynamic node scheduling. Both execute on real infrastructure, but each retains a persistent key holder: no decryption capability is partitioned across mutually distrusting nodes, and no share-level integrity check is provided.

Across these six studies, certified parameter generation, verifiable share distribution, threshold decryption, and containerised tenant isolation are each addressed in isolation. None binds verifiable secret sharing to a certificate-bearing Paillier key inside a deployed multi-tenant architecture, which is the gap addressed here.

2.1. Provable prime generation and Maurer's algorithm

Asymmetric schemes such as RSA, Diffie–Hellman and Paillier rest on large primes, and a composite factor invalidates the hardness assumption on which they depend. Conventional key generation establishes primality by probabilistic tests such as miller–Rabin and strong lucas, which bound the probability of accepting a composite but yield statistical confidence rather than proof; an undetected weak prime degrades the security of every operation performed under that key [17]. Maurer's recursive construction instead builds each prime together with a certificate [18], and Pocklington's theorem [19] verifies that certificate deterministically, which suits protocols requiring long-term trust in their keys.

2.2. Verifiable secret sharing

Shamir's (t, n) threshold scheme distributes a secret by polynomial interpolation, guaranteeing recoverability from any t shares and secrecy below that threshold [20], but it presumes an honest dealer. Verifiable secret sharing removes that assumption by letting each participant test its own share without learning the secret, and Pedersen's commitments bind the dealer to a single sharing polynomial while hiding the secret information-theoretically [9], [21]. Subsequent work has improved efficiency, resilience, and scalability: decentralised key generation [7], commitment-based homomorphic sharing that batches verification and lowers communication cost [14], and verifiable, accountable, and post-quantum variants [22], [23]. What remains unaddressed is Pedersen VSS applied to Paillier threshold decryption under provable-prime key generation [17], the gap this work fills.

3. METHOD

The framework treats number-theoretic guarantees, distributed key custody, and containerised deployment as one design. Paillier [24] parameters derive from Maurer-generated, Pocklington-certified primes, so every modulus factor carries a verifiable certificate and no probabilistic test is used. The decryption exponent is shared by Pedersen verifiable secret sharing, letting each participant validate its own share while no party holds the whole key. Evaluation uses five mutually distrusting Java containers over HTTP with a three-of-five threshold, making the cost of these guarantees measurable.

3.1. System architecture

In construction, the framework offers simultaneous guarantees over three essential properties: cryptographic safety, decentralized trust, and isolation of execution. Traditional approaches to instantiating Paillier run the entire cryptographic protocol stack inside a single trusted execution environment. This architecture breaks up the workflow into a coordinator/participant model and distributes computation of privacy-critical tasks to isolated containers. The architectural design draws on cryptographic protocol engineering. We visualize the proposed architecture in Figure 2. It consists of three key layers: a coordinator node (the coordinator node coordinates between participants), participant nodes, and a communication layer.

3.1.1. Coordinator node

The coordinator node performs two functions at a goal: It sets up the trusted cryptographic base and arbitrates among participants. It has primary responsibility for performing global actions and enforcing consistency of the system's state. It deterministically generates two 3072-bit primes $p \; and \; q$ from Maurer's recursive prime-generation algorithm and Pocklington certification as part of its initial setup. The coordinator node executes the Pedersen verifiable secret sharing procedure quickly. The secret key Λ is hardcoded as the

constant term of a degree-2 polynomial. Five shares are created as algebraic projections of the secret onto five independent, containerized participants.

3.1.2. Participant nodes

Each of the five participant nodes is a Java HTTP server running inside its own Docker container on an exposed port, so that no node can observe the internal memory or the runtime behaviour of another. Each node stores a single share S_i, delivered over an authenticated HTTP POST from the coordinator node. Shares are held in memory and never committed to disk, so a compromised node yields almost nothing to forensic analysis of its storage. Participant nodes never sign or verify on behalf of the system and never take part in reconstruction of the key, so no single node, however thoroughly compromised, holds enough information to reconstruct the private key.

3.1.3. Inter-container communication layer

The coordinator node and the participant containers communicate over TCP and HTTP, and only the coordinator node may initiate a transfer. Each participant container is exposed on a separate network port, giving a deterministic communication topology {8001, 8002, 8003, 8004, 8005} with no broadcast messages and no shared-memory side channels between participants. All exchanges are point-to-point and independent, and participants never interact with one another. During reconstruction, the coordinator node contacts the participants in turn over the same HTTP interface and uses the first three valid responses. The arrangement reflects the operating constraints of a genuine multi-tenant cloud platform, in which participants exchange information through structured APIs rather than through implicit trust in shared memory.

3.2. Coordinator node roles: Dealer and combiner

The coordinator performs two functions with different secret requirements, and the architecture separates them into roles with disjoint lifetimes. The dealer D is active only during setup. It generates the certified primes, derives Λ and the distributed exponent d, samples the sharing and blinding polynomials, publishes the commitments and verification keys, and transmits one share pair to each participant. It then erases p, q, Λ, d, all coefficients, all share pairs, and both Pocklington certificates, and terminates.

The combiner K is instantiated after dealing and persists thereafter. It requests partial decryptions, verifies each proof, combines the verified partials under integer Lagrange weights, and recovers the plaintext. Its working set public key, v, Δ, commitments, verification keys, ciphertext, and partials are entirely public, so K holds no secret at any point. Both roles run from the same container image: D as a setup container that exits when dealing completes, K as a long-running service. The separation confines the trust assumption to D and to the erasure window, which is what makes an honest-but-curious coordinator defensible in the security analysis.

3.3. Bulletin board

The hub-and-spoke topology is point-to-point, so a participant cannot make a statement the other participants can observe. The complaint protocol requires this: a complaint relayed by the coordinator can be discarded by a dealer that is itself the subject of the complaint. The architecture therefore adds an append-only bulletin board as a sixth container, holding the Pedersen commitments, the verification keys and any complaints. The dealer writes the commitments and keys once at setup; participants may append complaints, no party may modify or delete an entry, and all parties may read. Every participant then verifies its share against the same published commitments and observes every complaint, so a dealer distributing an inconsistent share is attributed and disqualified rather than merely detected locally. The board exposes one append and one read endpoint, holds a few kilobytes, and preserves isolation: participants still never address one another directly.

3.4. Stage I: certified prime generation

The security of the Paillier cryptosystem rests on algebraic properties of the modulus $n = pq$: if the primality of p or q fails, the hardness assumptions on which Paillier is built collapse. The framework therefore couples deterministic prime generation with a certificate of correctness, so that primality becomes a checkable assertion for every factor of the modulus. Where libraries normally rely on probabilistic tests, this work uses Maurer's recursive method with Pocklington certification to produce primes that are provably prime in the number-theoretic sense, building a large candidate around a smaller prime factor that is known in advance.

$$w = 2 \cdot R \cdot \rho + 1 \quad (1)$$

The structure gives the assurance that a large prime factor ρ of $w-1$ is known in advance, a requirement for determining the primality certification. The algorithm recursively generates ρ and R until the resultant answer w satisfies the bit-length requirement of $\lambda \geq 3072$ bits, at which point is accepted as a Paillier prime $(p, then\ q)$. The partial knowledge of the factorization of $w-1$ ensures a recursive approach and used in the Pocklington test. The Paillier modulus algebraic degeneracy is reduced by enforcing the constraints

$$gcd(pq, (p-1)(q-1)) = 1 \quad (2)$$

This condition makes for the structure's weakness that is likely to affect the HE decryption mapping by making sure that the modulus $n = pq$ doesn't share any non-trivial divisors with another component. Any candidate pair (p, q) that fails this criterion is discarded before further processing.

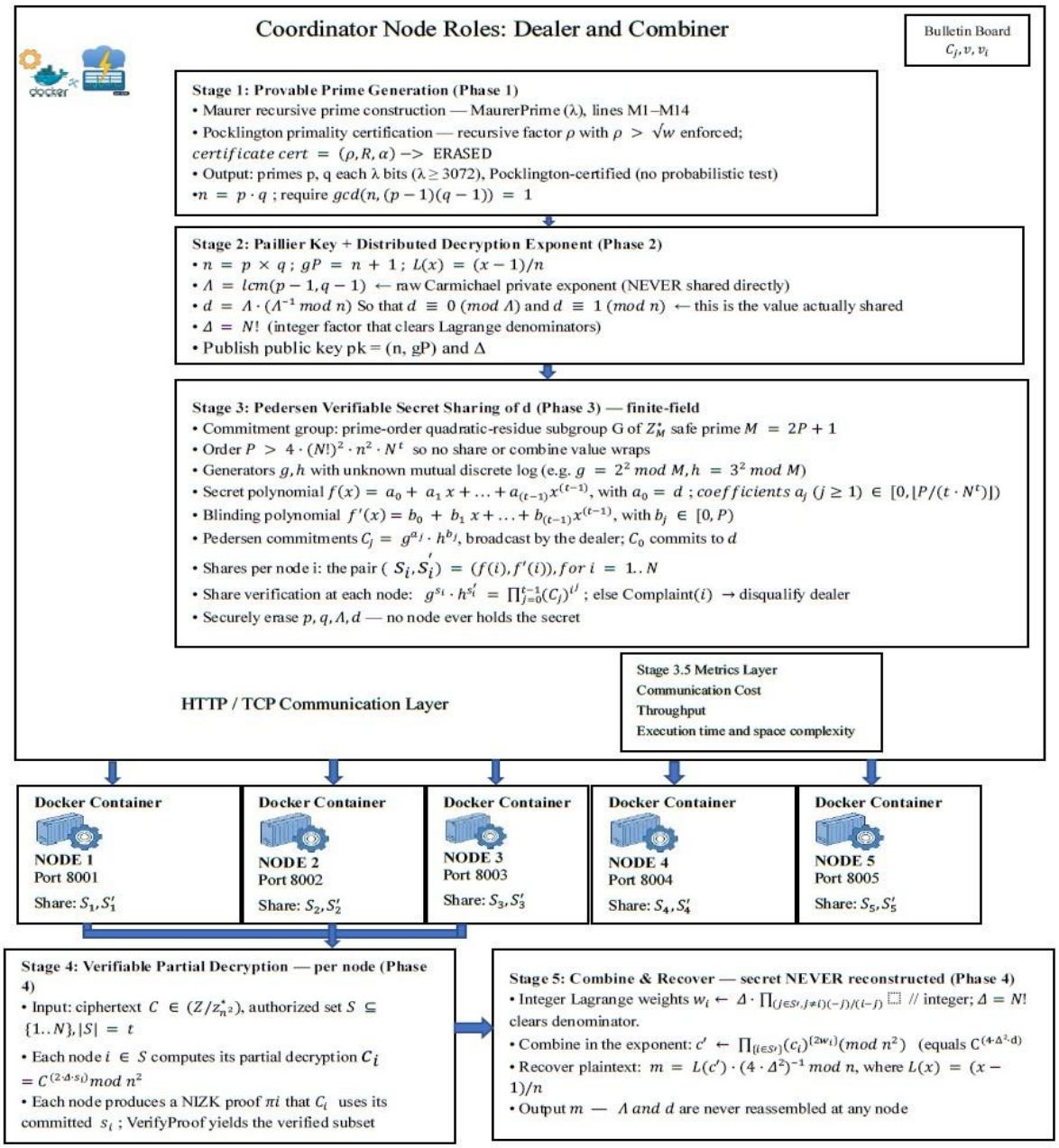


Figure 2. Stage-wise architecture of PPTP-VSS: coordinator dealer–combiner roles across Stages I–V, five containerised shareholders under a (3, 5) threshold, and the hub-and-spoke HTTP/TCP layer

3.4.1. Pocklington primality certification

When a candidate w is generated, it is subjected to primality determination certification via Pocklington's test. Let $a \in [2, \mathrm{w}-2]$ represent any witness randomly selected. This test confirms w as a prime number only if the condition is true:

$$a^{w-1} \equiv 1(mod\ w) \tag{3}$$

$$\gcd(a^{2R}-1, w) = 1 \tag{4}$$

The initial condition enforces Fermat's congruence, but the next condition guarantees that it is not divided by any non-trivial divisor that could affect the exponentiation. Due to the fact that ρ is a known prime factor of $(w-1)$ that exceeds $\sqrt{w}$ and$(w-1)/\rho = 2R$, the general Pocklington condition $\gcd(a^{2R}-1, w) = 1$ reduces to the form above. The triple (ρ, R, α) made up the Pocklington certificate. Primes that pass all checks are accepted, making sure that each generated prime is not merely provably prime but certified prime.

3.5. Stage II: Paillier key construction

Once p and q are certified, the framework derives the Paillier parameters, realising the Stage I guarantees as a working public-key scheme. Under the separation of duties imposed by the containerised design, every operation is carried out inside the coordinator node alone, so no participant container observes an intermediate value from which the private key could be recovered.

$$n = p \times q$$

3.5.1. Derivation of the private decryption exponent

The Carmichael formulation is used to compute the private decryption exponent Λ:

$$\Lambda = lcm(p-1, q-1) \tag{5}$$

The homomorphic decryption mapping is also modeled with the Carmichael formulation, which identifies the periodicity of the exponentiation group modulo n^2. Using the least common multiple rather than Euler's totient is important for the accuracy of the composite moduli and guarantees that the decryption function always runs smoothly on all valid ciphertexts. That fact that Λ is the raw Paillier private exponent; it is never shared directly; instead, the coordinator node derives the distributed decryption exponent

$$d \ = \ \Lambda \cdot (\Lambda - 1\ mod\ n) \tag{6}$$

This satisfies $d \ \equiv \ 0\ (mod\ \Lambda) and\ d \ \equiv \ 1\ (mod\ n)$. It is this value d, rather than Λ itself, that is subsequently distributed to the verifiable secret-sharing module. At this stage the full Paillier private capability of the system exists, but the framework forbids its storage or transmission in plaintext beyond the coordinator node.

3.5.2. Selection of the generator

The Paillier scheme requires a generator $g \in \mathbb{Z}_{n^2}^*$ satisfying the condition that the mapping $L(x) = \frac{x-1}{n}$ is invertible for gP^{Λ} mod n^2. The implementation selects

$$gP = n + 1$$

which satisfies Paillier's original construction and ensures that

$$L\ (gP^{\Lambda}\ mod\ n^2) \ = \ \Lambda\ mod\ n \tag{7}$$

while preserving full semantic security, this choice simplifies the encryption and decryption process.

3.5.3. Computation of the decryption coefficient

The coordinator Node computes the decryption coefficient

$$\mu \ = \ (L\ (gP^{\Lambda}\ mod\ n^2))^{-1}\ mod\ n \tag{8}$$

This quantity is used as a normalization constant for the decryption and guarantees that the decrypted plaintext can be properly re-mapped in the message space modulo n. The parameter μ is saved only inside the coordinator Node and is never shared externally with any participant node.

3.5.4. Public and private key separation

The cryptosystem is, at the end of this stage, a logical partition of: public key: (n, gP) private parameter: Λ, with auxiliary constant μ container-level computation; therefore uses the public key alone. For fragmentation, the private decryption exponent Λ is passed directly to the verifiable secret-sharing module; Λ is never transmitted unprotected, and Λ never exists as a whole outside the coordinator node.

3.6. Stage III: Pedersen verifiable secret sharing

The distribution phase converts the decryption exponent $d = \Lambda \cdot (\Lambda - 1 \, mod \, n)$ into a set of independently verifiable secret fragments, so that the decryption power of the system resides not in a single coordinator node but is spread cryptographically across mutually isolated execution contexts. The derived value is used because $d \equiv 1 \, (mod \, n) and \, d \equiv 0 \, (mod \, \Lambda)$. gives exactly the decryption capability required by the threshold protocol while the Carmichael value itself is erased. Sharing uses the Pedersen verifiable secret-sharing scheme instantiated over a finite-field prime-order group, so that every participant can check the consistency of its share against public commitments without learning the secret [25].

3.6.1. Commitment group

The commitment group is a prime-order quadratic-residue subgroup G of the multiplicative group Z_M^* where $M = 2P + 1$ is a safe prime and P is the prime order of G. The order P is chosen to satisfy the no-wrap bound $P > 4 \cdot (N!)^2 \cdot n^2 \cdot N^t$ such that throughout sharing or reconstruction, no share, commitment exponent, or combined value may ever encircle the modulus. We fix two generators of $G, g \, and \, h$, whose mutual discrete logarithm is unknown. They are regarded as the quadratic residues in the reference instantiation.

$$(g = 2^2 \, mod \, M, h = 3^2 \, mod \, M) \tag{9}$$

$g \, and \, h$ therefore generate the commitment group: squaring an element of Z_M^* forces it into the $order - P$ quadratic-residue subgroup. The two defining properties of a Pedersen commitment follow. Because $log_g h$ is unknown, the dealer is bound to a single sharing polynomial, while the secret remains hidden information-theoretically.

3.6.2. Sharing and blinding polynomials

To implement a $(t, N) = (3,5)$ threshold policy, the dealer constructs two polynomials of degree $t - 1 = 2$ over Z_P. The constant coefficient of the polynomial secret is the distributed exponent d:

$$f(x) = a_0 + a_1 x + a_2 x^2, a_0 = d \tag{10}$$

Secondly, a statistically independent blinding polynomial is drawn to mask the secret polynomial inside the commitments:

$$f'(x) = b_0 + b_1 x + b_2 x^2 \tag{11}$$

The non-constant secret coefficients $a_j \, (j \geq 1)$ are uniformly sampled from the reduced range floor $[0, \lfloor P/(t \cdot N^t) \rfloor]$, so that every polynomial evaluation is below P, and the blinding coefficients b_j are uniformly sampled from the full range [0,P). When $t - 1 = 2$, fixing the degree means that fewer than 3 evaluations give no information about d. Three different evaluations are enough to uniquely determine f and thus the secret.

3.6.3. Commitments to the coefficients

The dealer disseminates a Pedersen commitment for each coefficient index j, securing the secret coefficient a_j while completely concealing it with the blinding coefficient b_j:

$$C_j = g^{a_j} \cdot h^{b_j} \text{ (in G)} \tag{12}$$

In particular, the constant term commitment

$$C_0 = g^d h^{b_0} \quad (13)$$

is the public commitment to the secret d. The commitment $C_0 \dots \dots \dots C_{\{t-1\}}$ fixes the dealer to exactly one pair of polynomials (f, f'): because $log_g h$ is is unknown, the dealer cannot later open any commitment to a different value. Without this, the scheme would not actually be verifiable.

3.6.4. Generation and distribution of shares

For each container $i \in \{1, 2, 3, 4, 5\}$, the Coordinator Node examines both polynomials at the fixed non-zero point i to obtain a share pair:

$$(s_i, s_i') = \big(f(i), f'(i)\big)(mod\ P) \quad (14)$$

Each pair (s_i, s_i') is transmitted to node i over an authenticated, encrypted channel by a point-to-point HTTP POST initiated exclusively by the coordinator node. Shares remain in container memory and are never written to disk, so compromise of the storage layer yields nothing. Each participant receives only its own share and never observes the polynomials or the shares held by other participants; the flow is strictly one-directional.

3.6.5. Verifiable share consistency

When a node receives its share pair, it checks the share using the public commitments. It compares $g^{s_i} \cdot h^{s_i'}$ to the product $\prod_{j=0}^{t-1}(C_j)^{i^j}$. This way, each node can independently and quietly confirm its share fits with everyone else's, all without finding out the secret d or any of the polynomial's coefficients.

3.6.6. Complaint and disqualification protocol

If node i cannot verify its share it broadcasts a complaint (i) to the other participants. The participants then require the dealer to reveal the proof pair (s_i, s_i') for i. If the revealed pair does not verify, the dealer is disqualified and the protocol aborts; otherwise the share is valid, and execution continues. A dealer therefore cannot distribute inconsistent shares undetected, since any inconsistency is attributable and leads to disqualification.

3.6.7. Threshold security properties

The finite-field Pedersen sharing of d provides three properties:

- Confidentiality: Fewer than three share pairs reveal no information about d. Two colluding shareholders cannot recover the decryption exponent or the Paillier secret key.
- Correctness: Three valid share pairs determine the degree-2 polynomial f uniquely, and hence d, through integer Lagrange interpolation at the decryption threshold.
- Integrity: The published commitments prevent the dealer from acting maliciously. Tampering is detected either at share verification or later, when Paillier decryption fails, so any manipulation of the secret key becomes observable.

3.6.8. Secure erasure

Once dealing is complete the bridge node erases p, q, the Carmichael value and d; none is retained by any participant or device. What remains is five share pairs held in five separate Docker containers, together with C_0 through C_2. There is no central trust store, and the coordinator node alone can do nothing with the Paillier private key, which can be assembled only by a quorum under the (3,5) threshold. The private key is therefore not a static asset in a secret store but a capability constructed dynamically and auditably at the moment it is needed, realising the principle of least privilege for deployment in an untrusted cloud environment.

3.7. Stage IV: Verifiable threshold decryption

3.7.1. Decryption request and authorised set

As soon as the coordinator node receives ciphertext C, it starts decryption and selects a set S of exactly $t = 3$ nodes to do the request. It sends C to each node in that group:

$$C \in (Z/z_{n^2}^*), S \subseteq \{1..N\}, |S| = t \quad (15)$$

No assignment is fixed in advance: any three nodes that are online and responsive may serve. This reflects the conditions of multi-tenant infrastructure, in which containers may be unavailable or busy; all that

matters is that S can be formed with exactly t members. If such a set can be assembled the protocol proceeds, otherwise decryption is not performed.

3.7.2. Partial decryption

The value of each node i in S during this phase is only its share from Stage III. That is the only cryptographic value it carries around. After receiving C, the node computes a partial decryption at its site based on:

$$c_i = C^{(2\cdot\Delta\cdot s_i)} mod\ n^2 \tag{16}$$

Δ is the public integer $N!$, which removes the denominators of the Lagrange coefficients and is used again when the plaintext is recovered; the factor of two matches the $4\Delta^2$ normalisation applied at the end. Keeping $2 \cdot \Delta \cdot s_i$ in the exponent avoids fractions at every subsequent step, so the arithmetic remains integral and no value is ever reduced modulo the secret. The share s_i never leaves the node at which it was used; the only value transmitted is c_i, a group element, and c_i observed in isolation reveals little.

3.7.3. Proof of correct partial decryption

Partial decryption is only meaningful when all participants believe the partial decryption. Along with every c_i, node i also transmits an n-interactive zero-knowledge proof π_i , that indicates it computed c_i to the s_i value which it had already revealed during Pedersen sharing of in Stage III. The proof associates the current exponents to the public committed values previously revealed by the node. Therefore, no one can replace an invalid or cracked share in the partial decryption. The coordinator node examines every response by running VerifyProof(π_i), and retains only those nodes for which VerifyProof (π_i) evaluates to true. These constitute the confirmed set of nodes:

$$S' \subseteq S,\ |S'| \geq t$$

A node whose proof fails is discarded immediately and may be replaced. Verification proceeds as responses arrive and nothing is combined until the quorum is complete, which makes the injection of an incorrect contribution difficult to conceal.

3.7.4. Threshold enforcement

Because the sharing polynomial has degree $t - 1 = 2$, at least three valid partials are required to decrypt: the compromise of one or two containers reveals nothing about the plaintext, three verified partials suffice to begin recovery, and nodes outside the authorised set may remain offline without blocking the protocol, so the system tolerates $N - t$ unavailable nodes and never decrypts until a quorum of verified partials has been assembled. At the end of this stage the Coordinator Node holds only validated partials and their proofs, never a private key; these pass into the combine-and-recover phase of Stage V, where integer Lagrange weights bind them together in the exponent to yield the plaintext.

$$m = L(c') \cdot (4\Delta^2)^{-1}\ mod\ n\,,\ L(x) = (x - 1)/n \tag{17}$$

The Carmichael exponent is never rebuilt; it was erased in Stage III as soon as the shares were distributed. Decryption is performed jointly by a quorum, and it is at the point where the plaintext is produced that the distributed trust model matters most.

3.8. Stage V: combine and recover

Stage V returns the plaintext without the private key ever appearing intact. The coordinator node takes the validated partial decryptions from Stage IV and recovers the message by a single combination of the received group elements. Neither the Carmichael exponent nor d is ever recovered: the key remains split throughout, and the coordinator node works only with the product of group elements.

3.8.1. Ciphertext structure

The ciphertext of this scheme lies in the composite residue group modulo n^2 under the public key (n, gP). A message m, where m is drawn from $\mathbb{Z}_n$, is encrypted with a random nonce $r \in \mathbb{Z}_n^*$ as

$$C = gP^m \cdot r^n\ mod\ n^2 \tag{18}$$

The nonce r makes the encryption probabilistic: encrypting the same message twice yields two entirely different ciphertexts. This is also the basis of Paillier's additive homomorphism, under which operations on ciphertexts correspond to operations on the underlying messages.

3.8.2. Combining the verified partials

Recovery is performed entirely in the exponent. For each $i \in S'$ the coordinator node forms a unique integer Lagrange weight that maps the share i towards zero:

$$w_i = \Delta \cdot \prod_{(j \in S', j \neq i)(-j)/(i-j)} \quad \text{with } \Delta = \text{N!} \tag{19}$$

Multiplying by $\Delta = N!$ keeps every weight integral, since ordinary Lagrange interpolation would introduce fractions and $N!$ clears every denominator. The coordinator node then raises each c_i to twice its weight and multiplies the results. Substituting $c_\text{i} = C^{\{2 \cdot \Delta \cdot s_\text{i}\}}$ shows that the exponents sum to $4 \cdot \Delta^2 \cdot d$, so the distributed exponent d remains inside the exponent and is never seen or stored by the coordinator node.

$$c' = \prod_{\{i \in S'\}} (c_i)^{\{2w_i\}} (mod\ n^2) = \text{C}4 \cdot \Delta^2 \cdot \text{d} \tag{20}$$

The Paillier L function is then applied and the result multiplied by the modular inverse of $4\Delta^2$: Because $d \equiv 1\ mod\ n$, the message is preserved; because $d \equiv 0\ mod\ \Lambda$, the nonce is eliminated. $L(c')$ therefore holds $4\Delta^2 m\ mod\ n$, and dividing by $4\Delta^2$ through its modular inverse recovers m.

3.8.3. Correctness, ephemerality and container interface

Recovery is correct precisely when the verified set reaches the quorum $|S'| = t$: the weighted exponents then sum to $\Delta \cdot d$, giving $c' = C^{(4\Delta^2 d)}$, from which $L(c')$ isolates $4\Delta^2 m$ and hence m. Below the threshold, that exponent is never formed, and the recovered value is effectively uniform over $\mathbb{Z}_\text{n}$, so decryption is undefined. The capability is transient: the coordinator node retains c' and the plaintext only until the result is delivered and then erases both, while $p, q, \Lambda\ and\ d$ were destroyed at the end of Stage III and are never regenerated. The participant containers take no further part once they have returned their verified partials; the exponentiations, the integer Lagrange weights and the modular inverse are computed entirely within the bridge node, and no container observes c', the plaintext, or the contribution its own partial made.

3.9. Containerised deployment

A single declarative YAML file defines the entire deployment as a group of services, each specified by its build context, image, environment, network and runtime parameters, and launched and managed as one unit [26]. In this architecture that file holds one coordinator service running the coordinator logic and five shareholder services, node1 through node5, all launched from the same image and differentiated only by a runtime argument (["node1"]–["node5"]), which keeps dependency management and deployment declarative and reproducible.

3.10. Consolidated algorithm

Algorithm 1 generates a provable prime using the Maurer method and verifies its primality using the Pocklington criterion. Algorithm 2 presents the complete threshold Paillier process, including key generation, secret sharing, verification, and threshold decryption. This section presents the whole process as a single algorithm, from deterministic prime generation through to threshold-controlled Paillier decryption, including the coordinator-participant communication model and the containerised execution setup. Every symbol is defined where it is introduced in Sections 3.4 to 3.8.

Algorithm 1. MaurerPrime(b)

```
Require: bit-length b
Ensure: a provable prime w of exactly b bits, with Pocklington certificate cert
1:      if b ≤ 20 then      //base case: small certified prime
2:      repeat
3:      w ← random odd b-bit integer
4:      until w is prime by deterministic trial division
5:      return (w, ⊥)
6:      draw r from Maurer's size distribution, r ∈ (0.5, 1)
7:      bρ ← max(⌈r · b⌉, ⌊b/2⌋ + 1)      //factor exceeds half the bits, forcing ρ > √w
8:      (ρ, cert_ρ) ← MaurerPrime (b_ρ)      //recursively generated provable prime
9:      repeat
10:     R ← random integer with 2^((b−2)/ρ) ≤ R < 2^((b−1)/ρ)      //so w has exactly b bits
11:     w ← 2Rρ + 1
```

```
12:     α ← random base in [2, w − 2]   // Pocklington witness
13:     until w has no small prime factor, ρ² > w, α^(w−1) ≡ 1 (mod w) and gcd(α^(2R) − 1, w) = 1
14:     return (w, cert = (ρ, R, α))    // w − 1 = 2Rρ and ρ > √w, so w is certified prime
```

Algorithm 2. Distributed-trust threshold paillier

```
Require: security parameter λ (≥ 3072), node count N, threshold t ≤ N
Ensure: public key pk = (n, gP), public Δ and commitments C_0, …, C_{t−1}; node i privately holds
(s_i, s_i′); and plaintext m of any submitted ciphertext
        Phase 1 — Provable-prime generation (Maurer + Pocklington)
1:      repeat
2:      (p, cert_p) ← MaurerPrime(λ) // Pocklington-certified prime
3:      repeat (q, cert_q) ← MaurerPrime(λ) until q ≠ p
4:      n ← p·q
5:      until gcd(n, (p − 1)(q − 1)) = 1
        Phase 2 — Paillier key and distributed decryption exponent
6:      Λ ← lcm(p − 1, q − 1); gP ← n + 1; Δ ← N!
7:      d ← Λ · (Λ^(−1) mod n)    // d ≡ 0 (mod Λ) and d ≡ 1 (mod n)
8:      choose a group G of prime order P > 4 · (N!)² · n² · N^t with generators g, h whose mutual
        discrete logarithm is unknown    // no-wrap bound
9:      publish pk ← (n, gP), Δ and G
        Phase 3 — Pedersen verifiable secret sharing of d
10:     a_0 ← d;    sample a_1, …, a_{t−1} ∈ [0, ⌊P/(t · N^t)⌋); sample b_0, …, b_{t−1} ∈ [0, P)    // blinding
        polynomial
11:     f(x) ← a_0 + a_1 x + … + a_(t−1) x^(t−1);    f′(x) ← b_0 + b_1 x + … + b_(t−1) x^(t−1)
12:     for j ← 0 to t − 1 do
13:     C_j ← g^(a_j) · h^(b_j) in G; broadcast C_j    // Pedersen commit; C_0 commits to d
14:     for i ← 1 to N do
15:     (s_i, s_i′) ← (f(i), f′(i))
16:     send (s_i, s_i′) to node i over an authenticated, encrypted channel; node i stores the pair
17:     for i ← 1 to N, at node i do    //verify share
18:     if g^(s_i) · h^(s_i′) ≠ ∏_{j=0}^{t−1} (C_j)^(i^j) then broadcast Complaint(i)
19:     for each Complaint(i) do    // complaint / disqualify
20:     the dealer reveals (s_i, s_i′)
21:     if the test of line 18 still fails then disqualify the dealer and abort
22:     securely erase p, q, Λ and d    // no node ever keeps the secret
        Phase 4 — Verifiable threshold decryption (per ciphertext)
23:     on input ciphertext C ∈ (Z/z*_(n²)) and authorised set S ⊆ {1, …, N} with |S| = t do
24:     send C to every node i ∈ S
25:     for each i ∈ S, at node i do    //partial decryption
26:     C_i ← C^(2·Δ·s_i) mod n²
27:     construct proof ( ∏_i);  send (c_i ∏_i)
28:     S′ ← { i ∈ S : VerifyProof(∏_i) = true }
29:   for each i ∈ S′ do w_i ← w_i ← Δ · ∏_{(j∈S′, j≠i)} (−j)/(i−j) 1    // integer; Δ clears the
      denominators
30:     c′ ← ∏_{i∈S′} (c_i)^{2w_i} (mod n²)
31:     m ← L(c′) · (4 · Δ²)^(−1) mod n,   where L(x) = (x − 1)/n
32:     return m    // Λ and d are never reassembled at any node
```

3.11. Worked validation of the algorithm

- $N = 5$ node count (source value)
- $t = 3$ threshold; polynomial degree $t - 1 = 2$ (source value)
- $p = 7, q = 11$ smallest admissible paillier primes

$$n = p \cdot q = 77\,;\; n^2 = 5929$$

The recursive branch is illustrated by certifying $w = 11$ from $\rho = 5$, the machinery used at $\lambda \geq 3072$.

$$\rho = 5\,(base - case\ prime)\,,\; R = 1\,,\; w = 2 \cdot R \cdot \rho + 1 = 2 \cdot 1 \cdot 5 + 1 = 11\,,\; \alpha = 2$$

- $w - 1 = 10 = 2 \cdot R \cdot \rho$ (ρ divides $w - 1$)
- $\rho = 5 > \sqrt{11} = 3.317$ (pocklington size requirement)
- $\alpha^{(w-1)}\ mod\ w = 2^{10}\ mod\ 11 = 1$ (Fermat condition)
- $gcd(\alpha^{(2R)} - 1, w) = gcd(2^2 - 1, 11) = gcd(3, 11) = 1$ (no proper factor)

- Both conditions hold, so 11 is certified prime by $(\rho, R, \alpha) = (5, 1, 2)$, with no probabilistic test.
- Derivation of d: we need Λ^{-1} mod $n = 30^{-1} \, mod \, 77$. The extended Euclidean algorithm gives $18 \cdot 30 - 7 \cdot 77 = 1$, so $30^{-1} \equiv 18 \, (mod \, 77)$, and therefore

$$d = \Lambda \cdot (\Lambda^{-1} \, mod \, n) = 30 \cdot 18 = 540.$$

Order bound: $P > 4 \cdot (N!)^2 \cdot n^2 \cdot N^t = 4 \cdot 14400 \cdot 5929 \cdot 125 = 42\,688\,800\,000$. The smallest prime P above this bound whose safe-prime partner $M = 2P + 1$ is also prime was found by search:

$$P = 42\,688\,800\,509 \quad (prime)$$

$$M = 2P + 1 = 85\,377\,601\,019 \quad (safe \; prime)$$

$$g = 2^2 \, mod \, M = 4, \; h = 3^2 \, mod \, M = 9 \quad (QRs \Rightarrow order \; P; \; mutual \; log \; assumed \; unknown)$$

Test ciphertext. Encrypt $m = 50$ with randomness $r = 2$ $(gcd(2, 77) = 1)$:

$$C = gP^m \cdot r^n \, mod \, n^2 = 78^{50} \cdot 2^{77} \, mod \, 5929 = 1943.$$

Partial decryptions, $c^i = C^{\{2.\Delta.s_i\}} = C^{\{240.s_i\}}$:

$$c_1 = C^{131040} = 5391 \quad (240 \cdot 546 = 131040)$$

$$c_2 = C^{133440} = 2080 \quad (240 \cdot 556 = 133440)$$

$$c_3 = C^{136800} = 1002 \quad (240 \cdot 570 = 136800)$$

Integer lagrange coefficients, $w_i \leftarrow \Delta \cdot \prod_{(j \in S', j \neq i)(-j)/(i-j)}$

$$w_1 = 120 \cdot [(-2)/(1-2)] \cdot [(-3)/(1-3)] = 120 \cdot (2) \cdot (3/2) = 360$$

$$w_2 = 120 \cdot [(-1)/(2-1)] \cdot [(-3)/(2-3)] = 120 \cdot (-1) \cdot (3) = -360$$

$$w_3 = 120 \cdot [(-1)/(3-1)] \cdot [(-2)/(3-2)] = 120 \cdot (-1/2) \cdot (-2) = 120$$

Combine, $c' \leftarrow \prod_{\{i \in S'\}} (c_i)^{\{2w_i\}} (mod \, n^2)$ with exponents $2w_1 = 720, 2w_2 = -720, 2w_3 = 240$

$$c' = C_1^{720} . C_2^{-720} . C_3^{240} \, mod \, 5929 = 3543.$$

Recover, $m = L(c') \cdot (4\Delta^2)^{-1} \, mod \, n$:

$$L(c') = (3543 - 1)/77 = 46 \; (= 4\Delta^2 \cdot m \, mod \, n = 57600 \cdot 50 \, mod \, 77 = 46)$$

$$4\Delta^2 \, mod \, n = 57600 \, mod \, 77 = 4, \; 4^{-1} \, mod \, 77 = 58$$

$$m = 46 \cdot 58 \, mod \, 77 = 2668 \, mod \, 77 = 50 \; (= the \; encrypted \; plaintext)$$

The secret $d = 540$ is never reconstructed; only the combined exponent $4\Delta^2 d$ appears inside an exponent modulo n^2, and p, q, Λ, d were erased as soon as the dealing was completed.

4. RESULTS AND DISCUSSION

4.1. Experimental design

The experiment was designed to test the proposed cryptographic framework along the dimensions that determine whether it is viable in a multi-tenant containerised system: cryptographic correctness at every point of the protocol, enforcement of security through distribution and share integrity, and operational viability under container isolation. Specifically, the experimental design aims to verify that: i) provable prime

generation yields mathematically certified parameters for Paillier encryption, and ii) the framework executes correctly when deployed over isolated containers communicating through explicit network channels.

4.1.1. Execution environment

Each experiment was run on a commodity 64-bit x86 laptop running Linux 6.18.12+kali-amd64, with an Intel Core i7-4610M at 3.00 GHz and a 500 GB hard drive, using kernel-level containerisation through docker, which avoids the performance cost of full virtualisation while preserving component separation. Development and execution were carried out in IntelliJ IDEA, whose Docker integration runs each service in its own container rather than on the host. The Java runtime is provided by the Eclipse Temurin JDK 17 base image, chosen for its stability and its cryptographic provider interfaces. No shared filesystem, inter-process communication or shared memory is permitted; only explicitly defined network interfaces are available, a constraint imposed deliberately in order to reproduce the isolation conditions of a multi-tenant cloud deployment.

4.1.2. Container topology

The experimental architecture is deployed with Docker Compose as a six-container framework. The topology consists of:

- A dedicated coordinator container to establish cryptographic parameters, create orchestrations, and aggregate results.
- Five participant containers, one each for a unique tenant holding ownership and providing an individual share of the Paillier private decryption parameter.

Every participant container is built from the same Docker image and is specialised at runtime by command-line arguments that give it a unique logical identity and network port. The nodes therefore behave uniformly while remaining logically independent, and no request allows the coordinator container privileged access to participant state.

4.1.3. Network configuration and communication model

Containers communicate over lightweight TCP-based HTTP, each participant running a small server on a dedicated port that doubles as its node identifier, in a hub-and-spoke topology where the coordinator distributes share pairs and later collects partial decryptions, with no peer-to-peer traffic. This keeps message delivery and timing observable and controllable, permits explicit timeouts and connection limits that absorb network variance so that an unreachable node is simply marked unavailable, and mirrors production cloud key management, in which orchestration is centralised while confidential material stays dispersed.

4.2. Results for a 15-byte input

The normalised communication cost is defined as

$$\text{Communication Cost} = \frac{\text{Encrypted Data Size} - \text{Original Data Size}}{Original\ Data\ Size} \tag{21}$$

Table 1 summarises the simulation results for the 15 B input, giving the encrypted data size (1536 B), total execution time (679.41 s), throughput (0.0215 bytes/s) and communication cost (101.4).

Table 1. Simulation results for a data size of 15 B

File size (original data size)	Encrypted data size	Total execution time (sec)	Throughput (bytes/sec)	Communication cost
15Bytes	1536Bytes	679.413413938	0.021508046303986776	101.4

Table 2 reports the docker container statistics for the 15 B input across one coordinator and five participant nodes, covering CPU, memory, network and block I/O, and the number of processes running in each container. Memory consumption is low: node3 requires 37.33 MiB and node2 44.68 MiB, between 0.48 % and 0.57 % of the 7.667 GiB limit. Participant CPU utilisation lies between 0.04 % and 0.05 %, and each participant runs 21 or 22 processes. Block I/O varies, with reads ranging from about 86 kB to 18.3 MB, while writes never exceed 300 kB. The variation follows from some nodes reading more of the dataset than others. Network I/O is small, at 1.42 kB inbound with no outbound traffic. The coordinator container, which implements the coordinator node, is the busy node, requiring 100.34 % CPU and 438.2 MiB of memory, about 5.58 % of the limit, while the five participant nodes remain lightweight. The architecture therefore

places a negligible burden on overall resources, and the per-container footprint does not grow appreciably with data size. Figure 3 presents the simulation output for the 15 B input, confirming successful threshold decryption and message reconstruction.

Table 2. docker statistics showing space complexity for a data size of 15 B

Container ID	Name	CPU %	MEM usage / limit	MEM %	Net I/O	Block I/O	Pids
820abe8aabdd	node1	0.05%	40.83MiB / 7.667GiB	0.52%	1.42kB / 0B	86kB / 291kB	22
eadd90a9d63d	node2	0.05%	44.68MiB / 7.667GiB	0.57%	1.42kB / 0B	18.3MB / 188kB	21
77ceadb40cef	node3	0.05%	37.33MiB / 7.667GiB	0.48%	1.42kB / 0B	10.5MB / 180kB	21
180263babc33	node4	0.04%	40.22MiB / 7.667GiB	0.51%	1.53kB / 0B	13.2MB / 188kB	21
964a0467ab1e	node5	0.04%	43.25MiB / 7.667GiB	0.55%	1.42kB / 0B	16.5MB / 188kB	21
6aceb756d9ee	coordinator	100.34%	438.2MiB / 7.667GiB	5.58%	1.12kB / 0B	86kB / 291kB	21

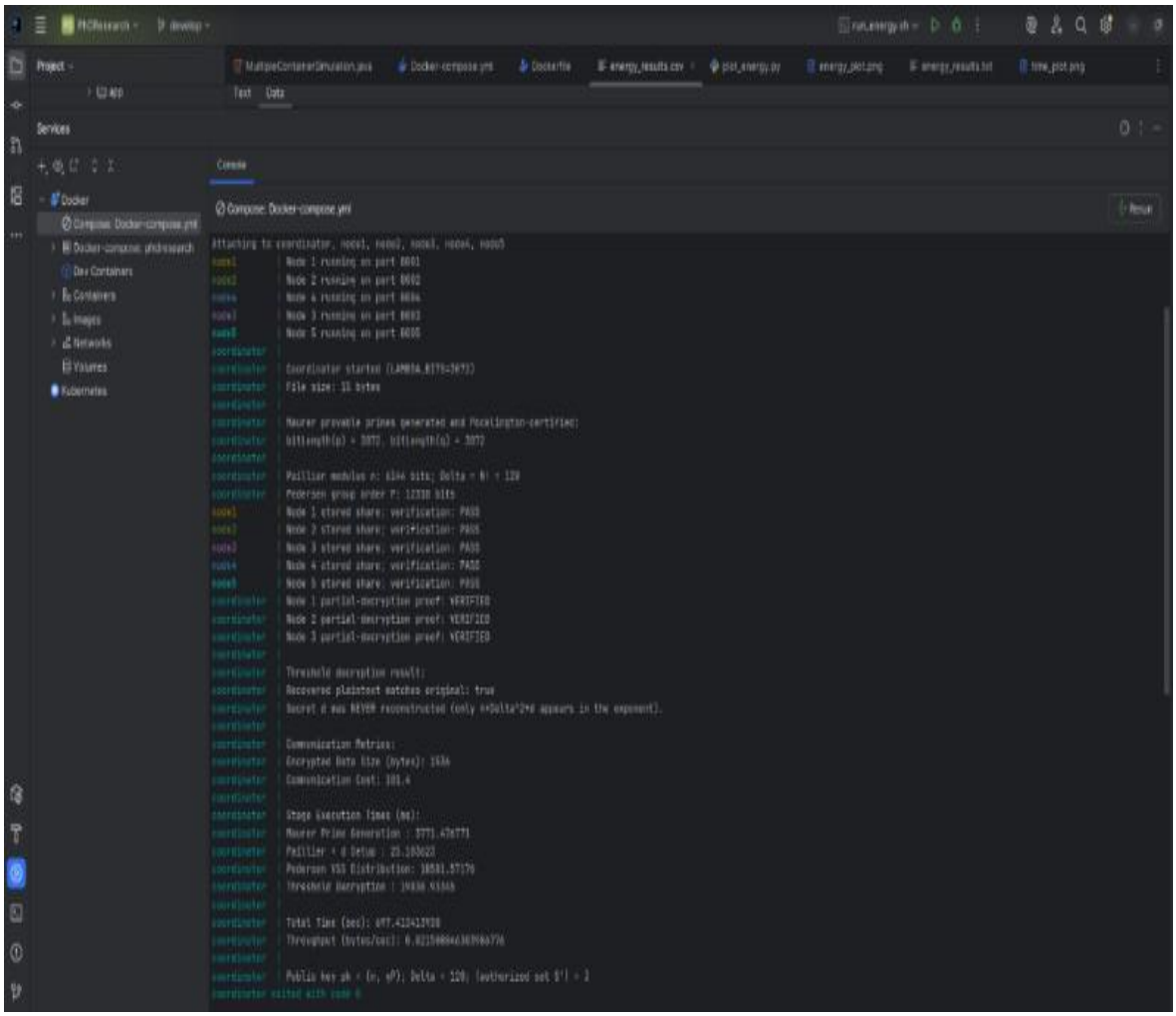


Figure 3. Simulation output of the proposed architecture for an input data size of 15 B

4.3. Comparative analysis

Of the four studies compared in Table 3, only the proposed PPTP-VSS combines certificate-bearing prime generation with verifiable (3, 5) threshold decryption, and it is the only one to report bytes-per-second throughput, a normalised communication cost, and measured per-node memory, CPU, and process footprints from an executed containerised deployment. P1 validates a keyless primitive by statistical testing alone, P2 optimises prime generation for speed without emitting a primality certificate, and P3 reports its metrics from NS-2 simulation only. The proposed framework is therefore the most completely evaluated of the four, and the only one whose key generation is deterministic rather than probabilistic.

Table 3. Comparative performance analysis of the proposed architecture against three recently published articles (2024-2025)

Dimension	This work (2026) Proposed PPTP-VSS	P1 – IJEEC 2025 [11]	P2 – PLoS ONE 2024 [17]	P3 – Sci. Reports 2025 [27]
Core approach	Provable-prime paillier (maurer + pocklington) with finite-field pedersen VSS; (3, 5) threshold; docker-orchestrated multi-tenant cloud.	Hash function constructed over the residue number system; modular processing across independent residues.	Two prime-generation algorithms based on odd-composed number formulas.	FA-FEO: FHE combined with CP-ABE and KP-ABE plus fast-exponentiation optimisation.
Key-parameter (prime) generation	Maurer recursive construction with pocklington certificate. Every modulus factor provably prime; no probabilistic test. $\lambda \geq$ 3072-bit primes.	Not addressed. Uses RNS moduli sets, not certified large primes.	Generation optimised for speed and benchmarked against miller–rabin and three sieves. No primality certificate emitted.	Not addressed; primes assumed from library defaults.
Key custody and trust model	No party holds the full key after setup. Λ, d and both pocklington certificates erased at the end of dealing; the secret is never reconstructed, even during decryption.	Not applicable — keyless primitive.	Not applicable — no key management layer.	Centralised attribute authority; a single entity holds the decryption capability.
Share verifiability	pedersen commitments published by the dealer; each node verifies its own share; complaint and dealer-disqualification protocol; per-node NIZK accompanies each partial decryption.	Not applicable.	Not applicable.	None.
Threshold decryption	(3, 5) threshold. Recovery performed in the exponent under integer lagrange weights cleared by $\Delta = N!$ tolerates 2 unavailable nodes.	None.	None.	None — single-key decryption.
Evaluation environment	Real containerized deployment: 6 docker containers over HTTP/TCP, Intel i7-4610M, Linux 6.18.12, Eclipse Temurin JDK 17.	Statistical test suite only (Hamming weight, bit balance, avalanche). No deployed system.	Algorithmic benchmark on a single machine. No cloud or distributed deployment.	NS-2 network simulation only. No physical deployment.
Throughput	0.0215 B/s (15 B). Small absolute values reflect byte-scale payload under 6144-bit threshold operations.	Not reported in B/s.	Reported as primes generated per unit time, not B/s.	Relative improvement over baseline FHE/ABE in simulated links; not in B/s.
Communication cost	101.4 . Hub-and-spoke HTTP only; no peer-to-peer traffic.	Not applicable.	Not applicable.	Per-node latency and bandwidth reported; not normalized per operation.
Space complexity (per node)	Measured per container: 27.7–44.7 MiB (< 0.6% of a 7.667 GiB limit), CPU 0.04–0.09%, 21–22 PIDs. Sub-linear in data size.	Not reported.	Not reported per node.	A memory metric is given; no absolute per-container footprint.
Multi-tenant isolation	Five mutually isolated containers, one per tenant, each holding a single share in memory only. Strict hub-and-spoke; no peer-to-peer channel and no shared memory.	Not addressed.	Not addressed.	Not modelled; simulation uses a flat IoT sensor network.
Reproducibility of results	Deterministic container images with fixed port topology; certified primes make key generation reproducible rather than probabilistic.	Test methodology described; no artefact released.	Algorithms fully specified; no artefact released.	Simulation parameters described; NS-2 scripts not released.

5. CONCLUSION

PPTP-VSS treats cryptographic correctness, distributed key custody and execution isolation as one design problem. Both factors of the Paillier modulus are Maurer-generated and Pocklington-certified, so every $\lambda \geq$ 3072-bit prime carries a certificate and no probabilistic test is used. The Carmichael exponent is never distributed; a derived exponent is shared by finite-field Pedersen VSS over a prime-order quadratic-

residue subgroup meeting an explicit no-wrap bound, so each node verifies its share and an inconsistent dealer is attributed and disqualified. Decryption runs as a (3, 5) operation entirely in the exponent under integer Lagrange weights cleared by $\Delta = N!$, so the private exponent is never reassembled. Six Docker containers on a 15-byte input decrypted correctly in 679.41 s at a normalised communication cost of 101.4, participants using 37.33–44.68 MiB against the coordinator's 438.2 MiB. Cost concentrates at the dealer–combiner; per-tenant isolation is nearly free. Throughput reflects a byte-scale payload under a 6144-bit modulus a deliberate trade-off, not a performance claim, and scaling remains unestablished. Future work will address larger payloads, fault injection of the complaint protocol, a formal proof binding partials to commitments, and distributed key generation and energy measurement.

ACKNOWLEDGMENTS

Not applicable

FUNDING INFORMATION

The author(s) received no financial support for the research, authorship, and/or publication of this article.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Michael Ayitey Junior	✓	✓	✓	✓	✓	✓	✓		✓	✓				
Peter Appiahene		✓				✓			✓	✓	✓	✓		
Obed Appiah		✓		✓	✓					✓	✓	✓		
Kwabena Adu				✓	✓					✓	✓	✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

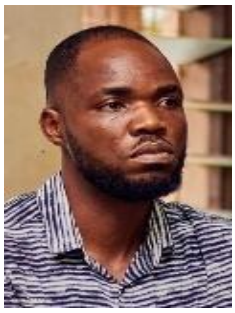
Authors state no conflict of interest.

REFERENCES

[1] Q. Zhou, H. Xie, and T. Su, "The impact of clock frequencies on remote power side-channel analysis attack resistance of processors in multi-tenant FPGAs," *Cryptography*, vol. 9, no. 1, p. 15, 2025, doi: 10.3390/cryptography9010015.
[2] H. Sturley, A. Fournier, A. Salcedo-Navarro, M. Garcia-Pineda, and J. Segura-Garcia, "Virtualization vs. containerization, a comparative approach for application deployment in the computing continuum focused on the edge," *Future Internet*, vol. 16, no. 11, p. 427, 2024, doi: 10.3390/fi16110427.
[3] Z. Zhang, Z. Xiong, and J. Ye, "Secure k-means clustering scheme for confidential data based on Paillier cryptosystem," *Applied Sciences (Switzerland)*, vol. 15, no. 12, p. 6918, 2025, doi: 10.3390/app15126918.
[4] D. Sousa-Dias, D. Amyot, A. Rahimi-Kian, M. Bashari, and J. Mylopoulos, "Cyclic homomorphic encryption aggregation (CHEA) a novel approach to data aggregation in the smart grid," *Energies*, vol. 17, no. 4, p. 878, 2024, doi: 10.3390/en17040878.
[5] S. Park, J. Lee, K. Harada, and J. Chi, "Masking and homomorphic encryption-combined secure aggregation for privacy-preserving federated learning," *Electronics (Switzerland)*, vol. 14, no. 1, p. 177, 2025, doi: 10.3390/electronics14010177.
[6] S. Park and J. Chi, "V-MHESA: A verifiable masking and homomorphic encryption-combined secure aggregation strategy for privacy-preserving federated learning," *Mathematics*, vol. 13, no. 22, p. 3687, 2025, doi: 10.3390/math13223687.
[7] K. W. Goh, B. U. I. Khan, A. R. Khan, D. S. Putra, F. R. Fathimathul, and M. A. Bhuyian, "Eliminating single points of trust: a hybrid quantum and post-quantum blockchain with distributed key generation," *Scientific Reports*, vol. 16, no. 1, p. 1286, 2026, doi: 10.1038/s41598-025-23310-6.
[8] J. Chen, X. Zhou, W. Fu, and Y. Mao, "Enhancing efficiency in trustless cryptography: An optimized SM9-based distributed key generation scheme," *Sensors*, vol. 24, no. 24, p. 7874, 2024, doi: 10.3390/s24247874.
[9] I. Alam, A. S. Alali, S. Ali, and M. S. M. Asri, "A verifiable multi-secret sharing scheme for hierarchical access structure," *Axioms*, vol. 13, no. 8, p. 515, Jul. 2024, doi: 10.3390/axioms13080515.

[10] M. Švarcmajer, M. Köhler, Z. Krpić, and I. Lukić, "Entropy extraction from wearable sensors for secure cryptographic key generation in blockchain and IoT systems," *Sensors*, vol. 25, no. 17, p. 5298, 2025, doi: 10.3390/s25175298.
[11] M. Pavlović, T. Talić, B. Damjanović, and N. Stamenković, "Residue number system-based hash function: Design and cryptographic analysis," *Ijeec - International Journal of Electrical Engineering and Computing*, vol. 9, no. 1, pp. 27–34, 2025, doi: 10.7251/ijeec2501027p.
[12] M. Malić, D. Dobrilović, D. Malić, and Ž. Stojanov, "Approach in the development of lightweight microservice architecture for small data center monitoring system," *Ijeec - International Journal of Electrical Engineering and Computing*, vol. 3, no. 2, pp. 61–69, 2020, doi: 10.7251/ijeec1902061m.
[13] Y. Tang, M. Guo, B. Li, K. Geng, J. Yu, and B. Qin, "Flexible threshold quantum homomorphic encryption on quantum networks," *Entropy*, vol. 27, no. 1, p. 7, 2025, doi: 10.3390/e27010007.
[14] S. Wang, C. Peng, X. Deng, Z. Peng, and Q. Chen, "Verifiable additive homomorphic secret sharing with dynamic aggregation support," *Electronics (Switzerland)*, vol. 13, no. 12, p. 2378, 2024, doi: 10.3390/electronics13122378.
[15] N. Shylashree and S. Kumar, "Homomorphic threshold detection for wireless sensor data using a privacy-preserving machine learning model," *Journal of Cloud Computing*, vol. 15, no. 1, p. 89, 2026, doi: 10.1186/s13677-026-00906-5.
[16] J. Wang and Y. Wang, "Privacy protection optimization method for cloud platforms based on federated learning and homomorphic encryption," *Sensors*, vol. 26, no. 3, p. 890, 2026, doi: 10.3390/s26030890.
[17] A. Ezz-Eldien *et al.*, "Computational challenges and solutions: Prime number generation for enhanced data security," *PLoS ONE*, vol. 19, no. 11, p. e0311782, 2024, doi: 10.1371/journal.pone.0311782.
[18] U. M. Maurer, "Fast generation of prime numbers and secure public-key cryptographic parameters," *Journal of Cryptology: The Journal of the International Association for Cryptologie Research (IACR)*, vol. 8, no. 3, pp. 123–155, 1995, doi: 10.1007/BF00202269.
[19] H. C. Pocklington, "The determination of the prime or composite nature of large numbers by Fermat's theorem," *Proceedings of the Cambridge Philosophical Society*, vol. 18, pp. 29–30, 1914.
[20] A. Shamir, "How to Share a Secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979, doi: 10.1145/359168.359176.
[21] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 576 LNCS, pp. 129–140, 1992, doi: 10.1007/3-540-46766-1_9.
[22] N. Kapalova, A. Haumen, and K. Algazy, "Secret sharing scheme with share verification capability," *Computers*, vol. 14, no. 9, p. 393, 2025, doi: 10.3390/computers14090393.
[23] J. Chen, H. Deng, H. Su, M. Yuan, and Y. Ren, "Lattice-based threshold secret sharing scheme and its applications: A survey," *Electronics (Switzerland)*, vol. 13, no. 2, p. 287, 2024, doi: 10.3390/electronics13020287.
[24] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," *Lecture Notes in Computer Science*, vol. 1592, pp. 223–238, 1999, doi: 10.1007/3-540-48910-X_16.
[25] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing paul feldman massachusetts institute of technology," *Network*, pp. 427–437, 1987.
[26] D. Moreau, K. Wiebels, and C. Boettiger, "Containers for computational reproducibility," *Nature Reviews Methods Primers*, vol. 3, no. 1, p. 50, Jul. 2023, doi: 10.1038/s43586-023-00236-9.
[27] K. D. More and D. Pramod, "A comparative performance analysis of fully homomorphic and attribute-based encryption schemes," *Scientific Reports*, vol. 15, no. 1, p. 35323, 2025, doi: 10.1038/s41598-025-19404-w.

BIOGRAPHIES OF AUTHORS



Mr. Michael Ayitey Junior is an Assistant Lecturer in the Department of Information Technology and Decision Sciences at the School of Sciences, University of Energy and Natural Resources. He holds a BSc and MPhil in Computer Science and is currently a Ph.D. candidate in Computer Science at the same university. His research interests include currency trend analysis using neural networks, image processing, and cloud security using cryptographic algorithms. He has six publications and was awarded Best Student Researcher in the Department of Computer Science and Informatics at the university's 2022 graduation ceremony. He can be contacted at email: michael.ayitey@uenr.edu.gh.



Prof. Peter Appiahene An Associate Proffessor with over 10 years of experience in academia and the computer science industry, both locally and internationally. He holds a PhD in Computer Science and is currently the Head of the Department of Information Technology and Decision Sciences at the University of Energy and Natural Resources, Ghana. His research interests include ICT for development, data science, machine learning, artificial intelligence, and digitization. his teaching areas cover algorithms and theory, artificial intelligence, data science, AR/VR, graphics and visualization, and information systems. He is a reviewer for international journals, including the Journal of Supercomputing, with several publications, and serves on the university's Governing Council and Electoral Commission of Ghana. He can be contacted at email: peter.appiahene@uenr.edu.gh.



Prof. Obed Appiah An Associate Proffessor at the University of Energy and Natural Resources, Ghana, with a Ph.D., MPhil, and BSc in Computer Science from Kwame Nkrumah University of Science and Technology. His research focuses on algorithm analysis, image processing, IoT, Industry 4.0, and machine learning. He teaches core computing courses and has extensive academic and industry experience. He supervises research, contributes to curriculum development, and has multiple publications in international journals, particularly in image processing and algorithm optimization. He can be contacted at email: obed.appiah@uenr.edu.gh.



Dr. Kwabena Adu is a Lecturer at the University of Energy and Natural Resources, Ghana, with a Ph.D. and MSc in Software Engineering from the University of Electronic Science and Technology of China and a BSc from the University of Education, Winneba. His research focuses on deep learning, artificial intelligence, and medical image analysis. He teaches, conducts research, and supervises students, with numerous publications in international journals and conferences, particularly in neural networks and medical image processing. He can be contacted at email: kwabena.adu@uenr.edu.gh.