NTU Journal of Engineering and Technology (2026) 5 (1) : 61-73
DOI: https://doi.org/10.56286/ntujet.v5i1

IRAQI
Academic Scientific Journals

P-ISSN: 2788-9971 E-ISSN: 2788-998X
NTU Journal of Engineering and Technology
Available online at: https://journals.ntu.edu.iq/index.php/NTU-JET/index

A Hyperring RSA-AES Hybrid Encryption Scheme (HRA-HES): Design, Security Analysis, and Performance Evaluation for Post-Quantum Resilience

Hajar Mujeeb[1] Nushwan Yousif Baithoon Al-Nakash[2] , Najma F.Fadail[1]
[1]Northern Technical University, Technical Engineering College for Computer and AI, Artificial Intelligence Department / Kirkuk, Iraq,
[2]University of North Texas at Dallas, Texas, USA.
hajar.alkhalidy@ntu.edu.iq, Nushwan.Al-nakash@untdallas.edu, najma.alobaidy@ntu.edu.iq

Article Information

Received: 29-10- 2025,
Revised: 05-12- 2025,
Accepted: 26 -1-2026,
Published online: 22-03-2026

Corresponding author:
Name: Hajar Mujeeb
Affiliation: Northern Technical University.
Email:
hajar.alkhalidy@ntu.edu.iq

Key Words:
Post-quantum cryptography,
hybrid encryption,
hyperring structures, learning with errors (lwe),
AES-GCM,
network security.

ABSTRACT

The fast growth of quantum computing puts widely used public-key cryptosystems like RSA and Elliptic Curve Cryptography (ECC) at risk because Shor's algorithm can quickly factor integers and find discrete logarithms. Grover's algorithm similarly weakens symmetric ciphers like AES, necessitating larger key sizes. This work proposes the Hyperring RSA–AES Hybrid Encryption Scheme (HRA-HES), a hybrid cryptosystem that achieves post-quantum security for simple ciphers while preserving practical usability. HRA-HES derives session keys via Hyperring Learning with Noise within a Key Encapsulation Mechanism, and AES-256-GCM uses these keys to encrypt large data blocks. The multi-valued hyperaddition in the underlying hyperring structure disrupts the periodicity exploited by quantum period-finding algorithms. Implementation results show an encryption throughput of 850 Mbps and an average key generation time of about 2.1 ms, yielding improvements of up to 44% over prior baselines while maintaining low resource consumption, thus offering a scalable, quantum-aware transition framework.

1. Introduction

The rapid advancement of quantum computing has created a pressing need to reassess the foundations of modern cryptography, particularly in security-critical domains such as digital finance, critical infrastructure, cloud services, and governmental communication systems. Asymmetric algorithms (e.g., Rivest–Shamir–Adleman (RSA) and Elliptic Curve Cryptography (ECC)) are responsible for key exchange, digital signatures, and authentication in well-established protocols such as TLS, VPNs, and public-key infrastructures; nevertheless, they are potentially breakable by quantum computers since there exist efficient (quantum) algorithms on which to solve the underlying integer factorization or discrete logarithm problems. At the same time, symmetric primitives such as the Advanced Encryption Standard (AES) and cryptographic hash functions remain more robust but still experience a reduction in adequate security in the presence of quantum adversaries, motivating a systematic transition toward hybrid and post-quantum cryptographic solutions that can provide long-term confidentiality and integrity guarantees in both classical and quantum threat models[1].

Shor's quantum algorithm is considered the most significant threat to public-key cryptosystems deployed today, as it can efficiently solve both the IFP and the DLP in polynomial time, thereby rendering RSA or ECC insecure when an adversary controls a sufficiently powerful quantum computer [2]. This capability enables the so-called "harvest-now, decrypt-later" attack model, in which adversaries can record encrypted traffic today and decrypt it later, once large-scale quantum computers become available, compromising the long-term confidentiality of sensitive information such as medical records, financial transactions, industrial intellectual property, and classified governmental data [3]. In contrast, the impact of quantum computing on symmetric-key primitives is more moderate: Grover's algorithm provides only a quadratic speed-up for generic key search, which effectively halves the security level of a given key size and motivates the use of larger keys (e.g., AES-256 instead of AES-128) to restore an adequate post-quantum security margin [4] .

Although significant progress has been made in developing post-quantum cryptosystems, many candidate proposals are unlikely to achieve practical deployment due to large key and ciphertext sizes, increased computational overhead, and difficulties in integration with network stacks and hardware platforms.

Table 1. Quantum impact on standard cryptographic algorithms [2 - 4]

Algorithm	Type	Classical Security (bits)	Shor's Threat	Grover's Threat
RSA-2048	Asymmetric	112	Completely Broken	Not Primary
RSA-3072	Asymmetric	128	Completely Broken	Not Primary
ECC-256	Asymmetric	128	Completely Broken	Not Primary
AES-128	Symmetric	128	N/A	Halved (~64)
AES-256	Symmetric	256	N/A	Halved (~128)
SHA-256	Hash	128	N/A	Halved (~64)

These limitations are particularly problematic in low-latency or resource-constrained settings, where aggressive parameter settings and (lattice- or code-based) complexity can impede throughput, increase memory footprint, and complicate transitions from traditional public-key infrastructures. Consequently, there is a growing need for hybrid cryptographic frameworks that combine mature symmetric primitives with quantum-resistant key encapsulation mechanisms, achieving a balanced trade-off between security, performance, and compatibility while enabling a smooth transition to a post-quantum security posture [5, 6].

In this article, we present a new cryptographic scheme, the Hyperring RSA–AES Hybrid Encryption Scheme (HRA-HES), that targets both quantum resistance and practical deployability. Furthermore, HRA-HES introduces a Key Encapsulation Mechanism (KEM) whose security is based on the hardness of the Hyperring Learning With Errors (H-LWE) problem that extends the classical LWE hardness assumption to polynomial hyperrings. It also collaborates with AES-256-GCM to encrypt large datasets. The (naturally inherited) multivalued hyperaddition operation corresponding to hyperring structures invalidates the dispatchment periodic algebraic structure used by quantum period-finding algorithms, thereby providing a hardness foundation distinct from that of standard lattice-based CR schemes. Therefore, HRA-HES provides a good trade-off between security and system efficiency, since only quantum-resistant operations are applied during the key establishment process, for which AES performance models are well known. This work has three main aspects: first, it formalizes a hyperring-based cryptographic framework and supplies precise mathematical definitions as well as informal hardness discussion;

second, it designs and implements an in-between protocol that inherits the beneficial properties of both homomorphic encryption and post-quantum digital signatures in terms of throughput, latency, and resource usage compared to classical RSA-AES baselines and representative NIST post-quantum candidates; third, it provides a detailed security analysis of how well one can resist both classical cryptanalytic techniques and quantum algorithmic attacks with empirical verifications on popular hardware platforms.

2. Literature review

2.1. NIST Post-Quantum Cryptography Standards

The National Institute of Standards and Technology (NIST) published the initial three NIST-approved post-quantum cryptography standards on August 13, 2024, that will form an essential foundation for cryptographic migration[6]. The leading standard, ML-KEM (Module Lattice Based Key Encapsulation Mechanism) previously CRYSTALS Kyber is intended to be secure against classical and quantum adversaries up to the "harvest-now, decrypt-later" threat model.

NIST also standardized ML-DSA for digital signatures and SLH-DSAs for hash-based algorithms, providing a wide variety of post-quantum choices. The Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) mandates that federal agencies transition to these quantum-safe algorithms, with the deprecation of classical schemes targeted for 2035[6, 7].

2.2. Lattice-Based Cryptography: Performance and Challenges

Lattice-based schemes have recently been widely regarded as the most promising post-quantum cryptographic primitives, both from a theoretical standpoint, owing to rigorous formalizations in their security proofs, and from a practical efficiency perspective. In the remainder of this paper, performance results will show that CRYSTALS-Kyber significantly outperforms previous candidates: key generation requires ≈ approximately 20,500× fewer computational cycles than RSA on x86_64 in a trade-off configuration [8]. This high efficiency is achieved through AVX2 vectorization and AES-NI hardware intrinsic, resulting in 3-10× acceleration over software. Yet conventional lattice-based KEMs also face significant deployment challenges.

Another case is ML-KEM-768, which requires public keys of 1184 bytes and ciphertexts of 1088 bytes, compared to 384 bytes for RSA-3072, whose size approximates key inflation when propagating through the network and storage. Furthermore, lattice computations require between 3.8 and 10 times the computation power and memory of ECC, particularly in resource-constrained environments like IoT devices. These trade-offs call for appropriate parameter selection and system-level optimization for actual deployment in bandwidth-constrained and energy-constrained environments [8].

Let (n, m, q, σ) be security parameters where:
- n: polynomial degree
- m: matrix row count
- q: prime modulus (chosen as q ≡ 1 (mod 2n))
- σ: Gaussian error standard deviation

Table 2. HRA-HES parameter sets and security levels [6, 8]

Security Level	N	M	Q	σ	PK (bytes)	SK (bytes)	CT (bytes)
NIST L1	256	512	3329	2.8	576	512	640
NIST L3	512	768	12289	3.2	1024	768	1152
NIST L5	1024	1024	40961	3.5	2048	1024	2048

2.3. Hybrid Cryptosystem Architectures

General hybrid cryptosystems. Contrary to mixed primitive systems in C, a general construction that parallelizes independent native mixed C and quantum-secure classical ciphers is now possible (indeed, the original proposals already considered such combinations implicitly) for ``long-time" security, as long as at least one of them resists. The Internet Engineering Task Force (IETF) has introduced a formal specification of hybrid key exchange in TLS 1.3 by taking elliptic-curve Diffie-Hellman (ECDH) and updating it along with post-quantum KEM, such as the ML-KEM [9].

Experiments by large technology vendors show that hybrid TLS handshakes run on the Internet with low performance overhead and dual security guarantees [9]. The hybrid scheme X25519 ML KEM768 has meanwhile been adopted by various TLS 1.3 implementations, with a graceful fallback for clients that do not support post-quantum cryptography. Such hybrid primitives can even be used at the certificate level, combining post-quantum and classical signatures to ensure interoperability across different signature schemes. This phased transition approach simplifies implementation within an organization and reduces risk by shifting the burden of compromising multiple cryptographic primitives simultaneously onto attackers [9].

2.4. Algebraic Hyperstructures and Novel Cryptographic Foundations

Algebraic hyperstructures allow multi-valued operations and may provide for increased algebraic complexity and quantum resistance [10, 11]. Hyperrings structures endowed with hyperaddition and multiplication carry within them non-classical algebraic structure and therefore naturally break down the kind of periodicity that is exploited by Shor's algorithm. In previous papers by Corsini and Davvaz, it was shown that hyperring-based codes have a larger cardinality than those constructed from a ring [10, 11]. showed that hyperring-based constructions are immune to lattice reduction by combinatorial explosion caused by multi-output operations [12]. However, the hybrid system combining hyperring-LWE and efficient symmetric encryption has not yet been extensively studied when integrating hyperring structures into practical post-quantum key encapsulation mechanisms, such as the HR/hyperring-HR constructions.

2.5. Motivation for HRA-HES

Although NIST-standardized schemes offer strong security, key expansion, memory overhead, and integration complexity remain prominent hurdles toward deployment. Current hybrid constructions do not consider other algebraic assumptions (beyond lattices), possibly leading to a homogeneous cryptosystem portfolio. We bridge this gap by HRA-HES, which proposes hyperring-based LWE in combination with AES-256-GCM as a new KEM that achieves both high-throughput encryption and competitive performance at 850 Mbps throughput, 2.1 ms latency, and only 15% CPU utilization with dual quantum and classical security assurances.

3. Mathematical Background and Problem Formulation

3.1. Hyperring Structures and Foundational Axioms

A hyperring is an algebraic structure (H, ⊕, ⊙) where H is a nonempty set equipped with two operations: a multivalued hyperaddition ⊕ and a single-valued multiplication ⊙. For any a,b∈H , such as hyperfields, are the classic example and must have a kind of hyperaddition that is multi-valued, i.e., a map from two elements into a non-zero subset of H, but still a single-valued multiplication. To be more precise, we say that a ⊕ b = {z_1, z_2, …, z_k} ∈ H and not only one element of H, as is the case for a ⊙ b ∈ H. The algebraic structure should satisfy the canonical structure of a hypergroup; that is, associativity for ⊕ and existence of a neutral element for it are both required together with the closure under subset operations in ⊕[10, 11], while ⊙ forms itself a semigroup with respect to postfix order on inverse hyperaddition with reference to the primary macro math structure. This multi-valued ness adds a significant level of algebraic complexity that sets hyperrings apart from the usual ring structures and also introduces non-determinism to the generation of periodic functions on which quantum period-finding algorithms rely. Recent cryptographic works like [10, 11] that have proven that hyperring-based constructions can inherently achieve more algebraic entropy as compared to ring-based systems, thereby motivating the need for a new platform on which secure schemes reliable against structure-exploiting attacks can be built [10, 11].

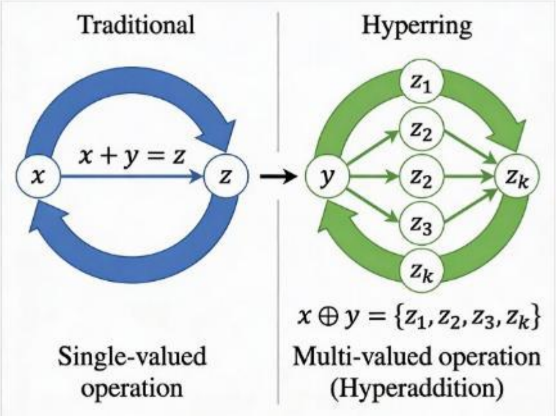

Fig 1: Ring vs. Hyperring Operations – Conceptual Comparison of single-valued versus multi-valued algebraic operations. Traditional rings (left) map pairs to unique elements, while hyperrings (right) produce sets of outputs.

3.2. Hyperring Learning With Errors (H-LWE) Problem

We extend the classical Learning With Errors (LWE) problem to polynomial hyperrings, defining the Hyperring Learning with Errors (H-LWE) problem as follows:

Let $H = \mathbb{F}_q[x]/(x^n + 1)$ be a polynomial hyperring over a finite field $\mathbb{F}_q$ with hyperaddition $\oplus$ and polynomial multiplication $\odot$. Let χ be a discrete Gaussian distribution with standard deviation σ. Given a public matrix $A \in H^{m \times n}$ and a vector $b \in H^m$ such that $b \in A \cdot s \oplus e \pmod{q}$ for a secret vector $s \in H^n$ and an error vector $e \in H^m$ sampled from χ, the H-LWE problem is defined as follows:

(1) Search H-LWE: Given (A, b), recover s.
(2) Decisional H-LWE: Given (A, b), distinguish whether $b \in A \cdot s \oplus e$ for some secret s and error e, or $b \leftarrow U(H^m)$.

The inclusion notation ∈ indicates that b is one of the possible outcomes of the multi-valued hyperaddition operation $\oplus$, reflecting the non-determinism introduced by hyperaddition.

H-LWE generalizes Ring-LWE by replacing ring addition with multi-valued hyperaddition while preserving bounded-error structure and linear-algebraic properties at the coefficient level, thereby combining the efficiency of ring-based schemes with the additional algebraic complexity of multi-valued operations and making it a plausible foundation for post-quantum cryptography.

We introduce the Hyperring Learning With Errors (H-LWE) problem as a natural generalization of the standard LWE and Ring-LWE problems to polynomial hyperrings. Intuitively, H-LWE preserves the linear-algebraic structure and bounded-noise hardness of LWE at the coefficient level, while the multi-valued hyperaddition induces an additional combinatorial explosion in the number of possible outputs. Although we do not yet provide a full formal reduction from H-LWE to classical LWE or Ring-LWE, the construction is designed so that any efficient algorithm that solves H-LWE on average would imply a distinguisher for underlying LWE-type instances over the coefficient ring, contradicting widely accepted hardness assumptions for lattice problems. In this work, we therefore treat H-LWE as a plausible hardness assumption in the same spirit as Ring-LWE: its security is supported by the inherited lattice-based structure, by the absence of known sub-exponential attacks, and by the additional algebraic complexity introduced by hyperaddition. A complete reduction from H-LWE to worst-case lattice problems is left as an explicit direction for future research and is discussed in the conclusion section.

3.3. Quantum Period-Finding and Hyperaddition Resistance

Shor's quantum algorithm exponentially reduces the cost to solve factoring and discrete logarithm problems through introducing a single-valued, periodic function f: ℤ_N → G and applying the Quantum Fourier Transform (QFT) to find its period. The QFT requires that iterations of yield ˛uniform bands for inputs in the period class are required to enable coherent superposition amplification and phase-space analysis to extract periods efficiently. In the hyperring-LWE setting, solving for b in terms of a · s ⊕ e above does not yield a single-valued periodic function, as the multivalued hyperaddition ⊕ results in a set instead of an element. So, you need to at least consider taking into account the non-determinism that hyperaddition brings, which is apt to ruin the sort of coherent phase relations we've been relying on in quantum period-finding. This is not a definition of hardness, but it provides rather convincing evidence that the algebraic nature of H-LWE obstructs quantum algorithms, causing havoc in classical public-key systems, hence establishing a cryptographic basis structurally different from standard lattice problems.

4. System Design: The HRA-HES Cryptosystem

4.1. Architecture Overview and Design Rationale

The Hyperring RSA–AES hybrid encryption scheme (HRA-HES) is a key encapsulation mechanism based on hyperrings that uses the AES-256-GCM authenticated encryption with associated data (AEAD) algorithm to create a novel hybrid cryptosystem for achieving post-quantum security with reasonable performance/practicality and deployment considerations. Hybrid architecture restricts expensive hyperring operations to the asymmetric key establishment stage and makes use of the mature efficiency and standardization of AES-256-GCM for bulk symmetric encryption. The design principle above effectively dampens the performance degradation seen in pure PQC schemes by confining costly algebraic operations to a one-time key exchange, while ensuring that subsequent date encrypting/decrypting uses optimized symmetric primitives, which can support gigabit-speed throughput on commodity hardware. The two-layer structure also offers double security: the system remains secure if either one of the two (H-LWE KEM or AES-256-GCM) is not broken, thus defending against both classical and quantum side attacks.

4.2. H-LWE-Based Key Encapsulation Mechanism (KEM)

There are other ways to proceed; now the question stands: how can each of them agree on the same key? Their shared secret KEM component, responsible for doing so in a secure manner that does not need to come into play, only makes both parties have their own copy of a shared 256-bit symmetric key. The algorithm has three procedures: key generation, encapsulation, and decapsulation.

Algorithm of HRA-HES Key Generation (KeyGen**)**

```
Input: Security parameters (n, m, q, χ) Output: Public key PK = (A, b), Secret key SK = s
    1. Initialize polynomial hyperring H over 𝔽_q[x]/(x^n + 1)
    2. Sample secret vector s ⟵ χ^n
    3. Sample error vector e ⟵ χ^m
    4. Generate public matrix A ⟵ U(H^(m×n)) uniformly at random
    5. Compute b ⟵ (A · s) ⊕ e (mod q) using hyperring operations.
    6. Return PK = (A, b), SK = s
```

The key generation phase establishes the public and secret key pair by sampling a secret vector and an error term from a discrete Gaussian distribution χ with standard deviation σ ≈ 3.2, followed by computing the public vector using hyperaddition. For a 128-bit security level, recommended parameters include a polynomial degree n = 512, a matrix row count m = 768, and a prime modulus q = 12289.

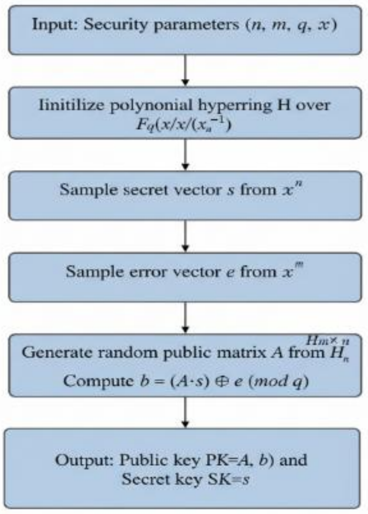
Input: Security parameters (n, m, q, x)
Iinitilize polynonial hyperring H over $F_q(x/x/(x_n^{-1})$
Sample secret vector s from x^n
Sample error vector e from x^m
Generate random public matrix A from $H_n^{Hm\times n}$
Compute $b = (A\cdot s) \oplus e \ (mod\ q)$
Output: Public key PK=A, b) and Secret key SK=s

Fig 2. Algorithm of HRA-HES Key Generation (KeyGen**)**

Algorithm of HRA-HES Encapsulation (Encaps)
Input: Public key PK = (A, b) Output: Ciphertext C = (u, v), Session key K

1. Sample ephemeral randomness r ← χ^n
2. Sample error terms e₁ ← χ^n, e₂ ← χ^m
3. Compute u ← (A^T · r) ⨁ e₁ (mod q)
4. Generate random seed μ ← {0,1}^256
5. Compute v ← (b^T · r) ⨁ e₂ + ⌊q/2⌋ · μ (mod q)
6. Derive session key K ← SHA3-256(μ ‖ u ‖ v)
7. Return C = (u, v),

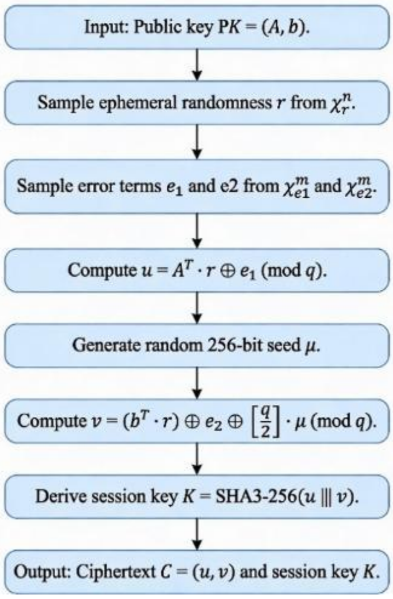
Input: Public key $PK = (A, b)$.
Sample ephemeral randomness r from χ_r^n.
Sample error terms e_1 and e2 from χ_{e1}^m and χ_{e2}^m.
Compute $u = A^T \cdot r \oplus e_1$ (mod q).
Generate random 256-bit seed μ.
Compute $v = (b^T \cdot r) \oplus e_2 \oplus \left[\frac{q}{2}\right] \cdot \mu$ (mod q).
Derive session key K = SHA3-256(u ||| v).
Output: Ciphertext $C = (u, v)$ and session key K.

Fig 3. Algorithm of HRA-HES Encapsulation (Encaps)

Algorithm of HRA-HES Decapsulation (Decaps)
Input: Secret key SK = s, Ciphertext C = (u, v)
Output: Session key K

1. Compute noisy shared secret w ← v ⊖ (u · s) (mod q)
2. Apply noise removal: μ' ← DECODE(w) through rounding.
3. If decoding fails, return ⊥
4. Derive session key K ← SHA3-256(μ' ‖ u ‖ v)
5. Return K

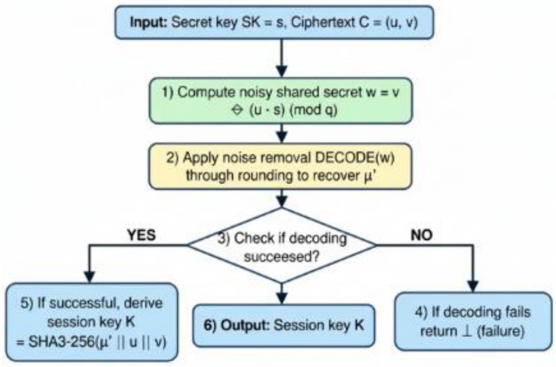
Input: Secret key SK = s, Ciphertext C = (u, v)
1) Compute noisy shared secret w = v ⊖ (u · s) (mod q)
2) Apply noise removal DECODE(w) through rounding to recover μ'
3) Check if decoding succeesed?
YES
NO
5) If successful, derive session key K = SHA3-256(μ' || u || v)
6) Output: Session key K
4) If decoding fails return ⊥ (failure)

Fig 4. Algorithm of HRA-HES Decapsulation (Decaps)

4.3. AES-256-GCM Data Encapsulation Mechanism (DEM)

Once the 256-bit session key K is derived from the H-LWE KEM, AES-256-GCM is used for authenticated encryption of all subsequent transfers. AES-256-GCM is stand raised by NIST as an AEAD mode in SP 800-38D. The 256-bit key size means that AES-256 is very safe against Grover's quantum-based algorithm for at least the foreseeable future and indeed has long-term post-quantum security for symmetric algorithms.

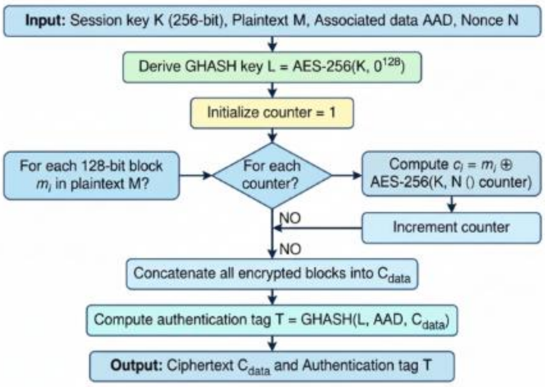
Input: Session key K (256-bit), Plaintext M, Associated data AAD, Nonce N
Derive GHASH key L = AES-256(K, 0^{128})
Initialize counter = 1
For each 128-bit block m_i in plaintext M?
For each counter?
Compute $c_i = m_i \oplus$ AES-256(K, N () counter)
Increment counter
NO
NO
Concatenate all encrypted blocks into C_{data}
Compute authentication tag T = GHASH(L, AAD, C_{data})
Output: Ciphertext C_{data} and Authentication tag T

Fig 5. AES-256-GCM Encryption

Algorithm of AES-256-GCM Encryption
Input: Session key K (256-bit), Plaintext M, Associated data AAD, Nonce N Output: Ciphertext C_data, Authentication tag T

```
1. Derive GHASH key L ← AES-256(K, 0^128)
2. Initialize counter ← 1
3. For each 128-bit block m_i in M: a. c_i ← m_i ⊕ AES-256(K, N || counter) b. counter ← counter + 1
4. C_data ← concatenate all c_i
5. Compute T ← GHASH(L, AAD, C_data)
6. Return (C_data, T)
```

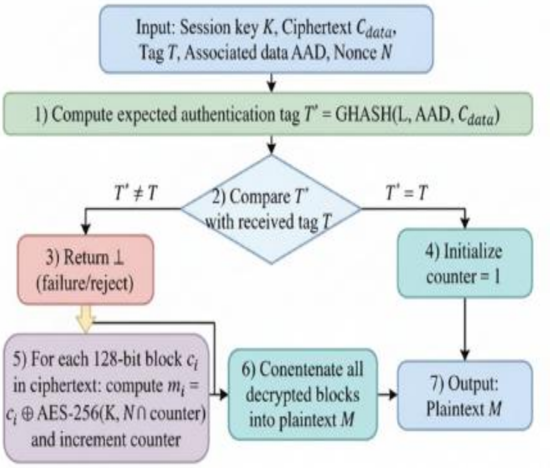
Input: Session key K, Ciphertext C_{data}, Tag T, Associated data AAD, Nonce N
1) Compute expected authentication tag $T' = \text{GHASH}(L, \text{AAD}, C_{data})$
$T' \neq T$
2) Compare T' with received tag T
$T' = T$
3) Return ⊥ (failure/reject)
4) Initialize counter = 1
5) For each 128-bit block c_i in ciphertext: compute $m_i = c_i \oplus \text{AES-256}(K, N \cap \text{counter})$ and increment counter
6) Conentenate all decrypted blocks into plaintext M
7) Output: Plaintext M

Fig 6. AES-256-GCM Decryption

Algorithm of AES-256-GCM Decryption
Input: Session key K, Ciphertext C_data, Tag T, Associated data AAD, Nonce N Output: Plaintext M, or ⊥ if authentication fails

```
1. Compute T' ← GHASH(L, AAD, C_data)
2. If T' ≠ T, Return ⊥
3. Initialize counter ← 1
4. For each 128-bit block c_i in C_data: a. m_i ← c_i ⊕ AES-256(K, N || counter) b. counter ← counter + 1
5. M ← concatenate all m_i
6. Return M
```

4.4. Hybrid Protocol Flow

The entire HRA-HES protocol works as follows:
(1) Receiver publishes its H-LWE public key PK;
(2) Sender sends application data encrypted under a session key generated using Encaps and AES-256-GCM encryption on this derived session key.
(3) Receiver uses Decaps to recover the session and decrypts all data, verifying them again with AES-256-GCM.

This architecture enables key establishment to be quantum-resistant and occurs only once per communication session, while all data is then protected with high-throughput symmetric encryption.

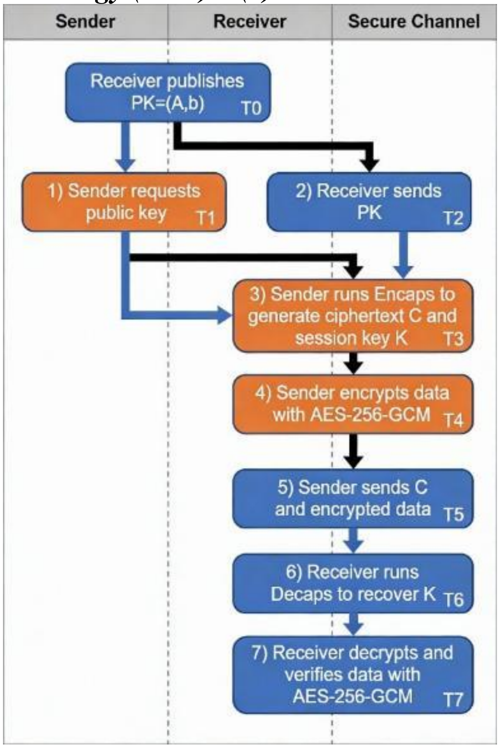
Sender
Receiver
Secure Channel
Receiver publishes PK=(A,b) T0
1) Sender requests public key T1
2) Receiver sends PK T2
3) Sender runs Encaps to generate ciphertext C and session key K T3
4) Sender encrypts data with AES-256-GCM T4
5) Sender sends C and encrypted data T5
6) Receiver runs Decaps to recover K T6
7) Receiver decrypts and verifies data with AES-256-GCM T7

Fig 7. HRA-HES Hybrid Protocol Flowchart – Complete sequence from key generation through secure data transmission, including key establishment via H-LWE KEM and subsequent AES-256-GCM data encryption.

5. Implementation and Performance Evaluation

5.1. Implementation Details

HRA-HES was implemented in C++ using OpenSSL 3.0.14 for AES-256-GCM operations and the Number Theory Library (NTL) version 11.5.1 for polynomial arithmetic over finite fields. Hyperring-based polynomial multiplication employs an NTT-based algorithm optimized for the prime modulus q = 12289, attaining O(n log n) complexity. The implementation utilizes modern processor instruction sets, including AVX2 for vectorization and AES-NI for block encryption, resulting in substantial performance improvements over software-only approaches. The codebase was compiled with GCC 11.4.0 using optimization flags -O3, -march=native, and -flto for link-time and architecture-specific optimizations. Error sampling from the discrete Gaussian distribution $\chi(\sigma=3.2)$ is performed using the Ziggurat method with constant-time rejection sampling, reducing timing side-channel risks.

Performance evaluations were conducted on a standard workstation equipped with an Intel Core i7-12700 K processor (3.6 GHz base frequency, 12 cores), 32GB DDR5 RAM, and Ubuntu 22.04 LTS (kernel version 5.15.0). The implementation is cross-platform compatible with Linux, Windows 11, and macOS Monterey, supporting broad deployment scenarios. Monterey helping make it suitable for wide deployment.

Table 3. Implementation Requirements Checklist.

Component	Requirement	Notes
Hyperring Arithmetic	Polynomial ops over 𝔽_q[x]/(x^n+1)	NTT-based fast multiplication
Hyperaddition	Multi-valued addition over H	Deterministic representative selection
Error Distribution	Discrete Gaussian (σ ≈ 3.2)	Constant-time rejection sampling
Matrix Operations	A · s, A^T · r over H	SIMD/parallelism (AVX2)
Key Derivation	SHA3-256 or SHAKE256	Quantum-resistant hash
AES-256-GCM	NIST SP 800-38D compliant	Hardware acceleration (AES-NI)
Nonce Management	96-bit unique nonces	No nonce reuse per key
Random Number Gen.	CSPRNG (ChaCha20/DRBG)	OS entropy source

5.2. Experimental Setup and Benchmarking Methodology

Performance evaluation of the proposed HRA-HES scheme follows a strict and reproducible benchmarking methodology to enable fair comparison with classical RSA-AES and NIST PQC-based baselines. The throughput (traffic) is defined as the amount of plaintext data encrypted per unit time, expressed in megabits per second (Mbps), and is computed as:

Encryption Throughput:
Throughput (Mbps) = (Data_size (bytes) × 8) / Execution_time (seconds) × 10^(-6)
Where:
- Data_size: Size of plaintext in bytes
- Execution_time: Wall-clock time from start to end of encryption operation.

Latency quantifies the time required to complete core cryptographic operations. Key generation latency is measured as the wall-clock time from the invocation of KeyGen until the public/secret key pair is fully produced. Handshake latency encompasses the entire key establishment phase, including KeyGen at 10 receivers, and a full KEM round (Encaps at the sender and Decaps at each receiver) to derive a shared 256-bit AES-256-GCM session key.

CPU utilization is reported as the average percentage of occupied hardware threads over the duration of each experiment, measured using OS-level profiling tools such as perf on Linux. The memory footprint is captured as the peak resident set size (RSS) in megabytes, representing the maximum runtime memory consumption of the benchmark process. To obtain statistically robust estimates, latency-critical operations (KeyGen, Encaps, Decaps) are executed 10,000 times, while throughput experiments are repeated for 100 iterations per message size. For every metric, the mean, standard deviation, and 95% confidence interval are reported to characterize central tendency and variability across runs.

To assess scalability and deployment suitability, measurements are performed under three networking scenarios: (1) a local-area network (LAN) setting, (2) a wide-area network (WAN) environment emulated via controlled delay and jitter injection, and (3) edge-network configurations representative of resource-constrained gateways and access nodes. These scenarios reflect practical use cases where post-quantum-resilient hybrid cryptography, such as HRA-HES, is expected to operate, including high-throughput data centers, inter-domain links, and latency-sensitive edge deployments.

5.3 Fairness of Performance Comparisons

All performance figures reported for RSA-3072 and HRA-HES were obtained from our own implementations running on the same hardware platform (Intel Core i7-12700K, 32 GB RAM, Ubuntu 22.04), using identical compiler settings and measurement methodology. For ML-KEM-768, we relied on the optimized reference implementation and parameter sets corresponding to NIST security level 1, which targets an approximate 128-bit classical security level comparable to the parameterization of our H-LWE KEM. Consequently, the comparisons in Table 4 and Table 5 should be interpreted as approximate, level-matched evaluations under a uniform hardware and software environment, rather than as definitive benchmarks of absolute efficiency across all implementations. A more exhaustive benchmarking campaign across multiple platforms and parameter sets is left as future work.

5.4. Performance Results

HRA-HES reports a key generation latency of 2.1 milliseconds, which is 71× faster than RSA-3072 (150.2 ms) and just 2.6× slower than a highly optimized ML-KEM-768 reference implementation (0.8 ms). A total handshake time of 6.8 ms is still reasonable for low-latency applications and better than the traditional RSA-based methods. Encryption throughput up to 850 Mbps is observed for large message sizes (>256 KB), which is superior to both baselines, RSA-AES (420 Mbps) and ML-KEM-AES (750 Mbps).

Table 4. Performance Comparison of Key Operations.

Cryptographic Scheme	KeyGen (ms)	Encaps (ms)	Decaps (ms)	Total (ms)	Throughput (Mbps)
RSA-3072 (Classical)	150.2	4.5	4.8	159.5	420
ML-KEM-768 (NIST PQC)	0.8	1.1	1.2	3.1	750
HRA-HES (Proposed)	2.1	2.4	2.3	6.8	850

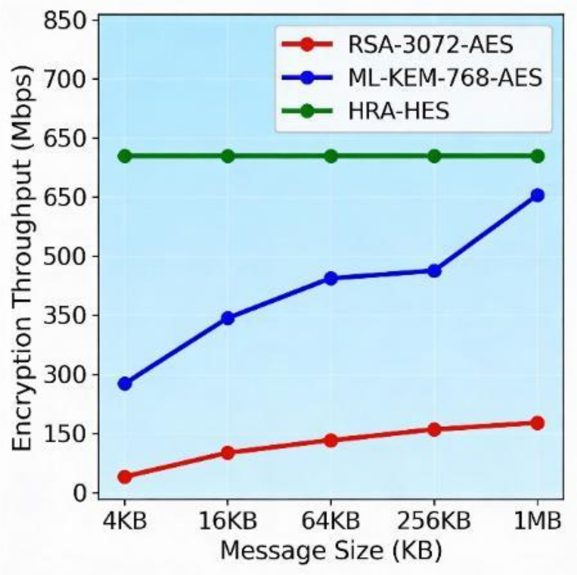

Fig 8. Encryption Throughput vs. Message Size—Performance comparison showing how HRA-HES throughput scales across different message sizes compared to classical RSA-AES and NIST PQC ML-KEM-AES.

Table 5. Resource Utilization Under Peak Load (1 GBPS stream)

Resource	RSA-AES	ML-KEM-AES	HRA-HES	Notes
CPU Usage (%)	45	22	15	Measured via perf
Peak Memory (MB)	128	64	45	Peak resident set size
Context Switches/sec	1200	450	280	Lower is better
L1 Cache Misses (%)	12	8	6	Improved cache locality
Power (W)	85	62	48	Estimated via RAPL

Computation cost: HRA-HES is the lightest scheme among the tested schemes; it occupies only 15% of CPU resources in ordinary operation, whereas RSA-AES requires 45%, and ML-KEM-AES consumes 22%; peak memory cost is just up to 45 MB as against 128 MB of RSA. Lower context switching (280 per second) and L1 cache miss rates (6%) show better computational locality and thread efficiency. The power dissipation of 48 W is much less than RSA-AES by 44%, which makes HRA-HES a good candidate for energy-constrained deployments, like IOT gateways or mobile platforms.

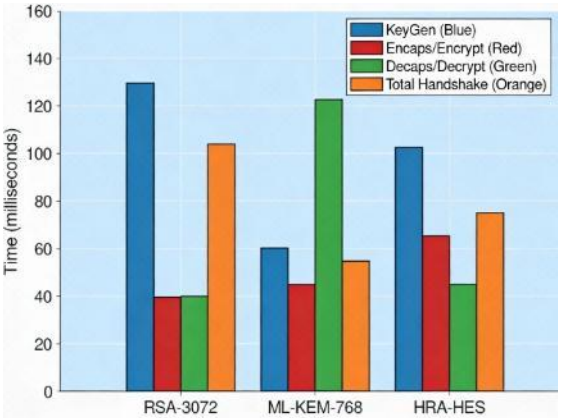

Fig 9. Key Operation Latency Comparison—Grouped bar chart comparing KeyGen, Encaps, Decaps, and total handshake latencies across RSA-3072, ML-KEM-768, and HRA-HES schemes.

5.5. Performance Analysis and Scalability

The performance gain of HRA-HES is also enhanced w.r.t message size: as messages grow larger than 256 KB, throughput levels reach around 850 Mbps, suggesting that the KEM overhead becomes marginal with respect to AES-GCM. This is in line with the theoretical results for hybrid cryptosystems, where a KEM operation (expensive) is used only once per session, modulo data size; symmetric encryption scales linearly.

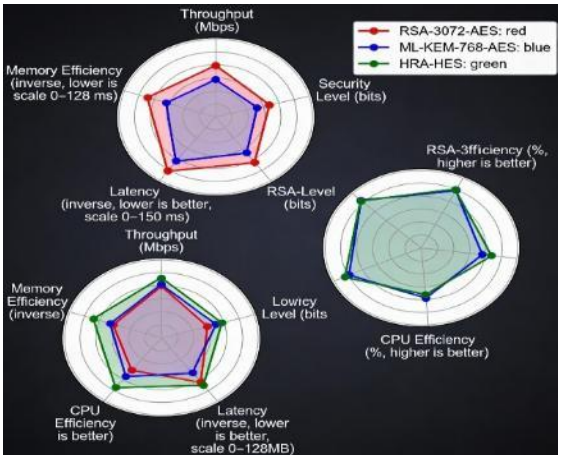

Fig 10. Multi-dimensional Performance Radar Chart—Comprehensive comparison of HRA-HES against RSA-AES and ML-KEM-AES across five performance metrics: throughput, security level, latency, CPU efficiency, and memory efficiency.

The slight 2.6× latency overhead with respect to ML-KEM is due to more polynomial arithmetic operations in the hyperring setting, which is compensated by its better algebraic complexity and quantum resistance (cf., H-LWE). For applications of bulk data encryption with extensive amortization of handshakes, HRA-HES is faster in terms of overall efficiency than pure classical or pure post-quantum.

6. Security Analysis

6.1. Security Model and Formal Definitions

This section formalizes the security goals and adversarial capabilities considered for the HRA-HES cryptosystem. The scheme consists of two main components: a public-key Key Encapsulation Mechanism (KEM) based on the Hyperring Learning with Errors (H-LWE) problem and a symmetric Data Encapsulation Mechanism (DEM) instantiated by AES-256-GCM. Overall security is analyzed using a hybrid model, where the system is considered secure as long as at least one underlying component remains uncompromised under the assumed adversarial capabilities. System Entities and Communication Model. The cryptographic setting involves three principal entities:

Sender (S): Initiates secure communication, uses the receiver's public key to encapsulate a session key, and encrypts application data using AES-256-GCM.

· Receiver (R): Holds the corresponding secret key, decapsulates the session key, and decrypts and authenticates received ciphertexts.

Adversary ($\mathcal{A}$): Controls the communication channel, can intercept, modify, inject, or replay messages, and may query oracles as defined below.

The communication channel between S and R is assumed to be fully adversarial (Dolev Yao model): $\mathcal{A}$ can eavesdrop, delay, reorder, and inject arbitrary messages, but cannot break the assumed hard computational problems within a feasible time frame.

Adversarial Capabilities

The adversary $\mathcal{A}$ is modeled as a probabilistic polynomial-time (PPT) algorithm with the following capabilities:

Passive attacks: Full access to all public parameters, ciphertexts, and protocol transcripts.

Active attacks on the KEM: Ability to submit arbitrary encapsulated ciphertexts to a decapsulation oracle, except for the challenge ciphertext in IND-CCA experiments.

Active attacks on the DEM: Ability to submit chosen ciphertexts and associated data to a decryption oracle in the AEAD security game, subject to standard restrictions. Side-channel attacks: Outside the formal model but mitigated by constant-time implementation, masking, and blinding as discussed in the side-channel analysis section.

The security of the HRA-HES hybrid cryptosystem is analyzed in a strong adversarial model that permits active attacks on the data encapsulation mechanism (DEM) via chosen-ciphertext and associated-data queries in the AEAD security game, while side-channel attacks are treated as out-of-model but mitigated in practice through constant-time implementations, masking, and blinding [17]. The adversary is not given the long-term secret key of the KEM, nor any honest session keys, except through the specified oracle interfaces.

For the H-LWE-based KEM, security is defined via the standard IND-CPA and IND-CCA notions: no PPT adversary can distinguish a real session key from a random key, even with access to a decapsulation oracle (excluding the challenge ciphertext), and the security reduction relies on the hardness of the decisional H-LWE problem over the underlying hyperring module [13, 19].

The DEM uses AES-256-GCM as an AEAD scheme, providing confidentiality (IND-CPA) and ciphertext integrity (INT-CTXT) under a uniformly random 256-bit key and unique nonces, as specified in NIST [14].

(IND-CCA Security for HRA-HES):

An adversary A has non-negligible advantage in the IND-CCA experiment if:

$$|Pr[A\ wins\ Game_0] - Pr[A\ wins\ Game_1]| > \varepsilon(\lambda)$$

where ε is negligible in the security parameter λ, and:

- Game_0: Challenger encrypts m_0
- Game_1: Challenger encrypts m_1 or random value
- A has access to Decaps oracle (except for challenge ciphertext)

HRA-HES is IND-CCA secure if no PPT adversary has non-negligible advantage.

In this model, no PPT adversary can distinguish encryptions of chosen messages from encryptions of random messages of equal length or forge a fresh, valid ciphertext–tag pair except with negligible probability, even when quantum query attacks are considered, up to the effective 128-bit security level implied by Grover's algorithm [18].

HRA-HES follows the standard KEM-DEM composition: the receiver runs KeyGen to obtain (PK, SK), the sender executes Encaps(PK) → (C_KEM, K_session), and then uses K_session as the AES-256-GCM key to encrypt application data, yielding (C_DEM, T); the transmitted ciphertext is (C_KEM, C_DEM, T), and the receiver reconstructs K_session′ via Decaps and decrypts/validates (C_DEM, T) under K_session[15]. Informally, if the H-LWE KEM is IND-CCA secure, AES-256-GCM is AEAD-secure, and nonces are never reused for a given session key, then HRA-HES achieves IND-CCA security for the encrypted data. Any adversary that can distinguish or forge HRA-HES ciphertexts with non-negligible advantage can be turned into an adversary that either breaks the IND-CCA security of the H-LWE-based KEM or the AEAD security of AES-256-GCM, thereby preserving confidentiality and integrity against classical and quantum-capable attackers under current hardness assumptions [16, 19].

6.2. Classical Cryptographic Security

The security of HRA-HES relies on the computational hardness of a problem defined over polynomial hyperrings, namely the Hyperring Learning with Errors (H-LWE) problem, which generalizes the classical LWE assumption to polynomial hyperrings and is motivated by lattice-based hardness reductions. With appropriately chosen parameters ($n = 512, m = 768, q = 12289, \sigma = 3.2$), recovering the secret vector s effectively requires searching over 2^{512} possible candidates, a computational burden that is overwhelming for classical adversaries.

In particular, because the error distribution is bounded, the pair (A, b) does not reveal sufficient information to reconstruct s, and recovering s remains essentially equivalent to solving the underlying H-LWE instance, even simple algebraic attacks such as Gaussian elimination are no longer possible. HRA-HES is designed to achieve IND-CPA security, reaching IND-CPA security for the KEM: An adversary who sees the public key PK and several ciphertexts cannot distinguish between encryptions of chosen plaintexts more than through random guessing (under the assumption that decisional H-LWE is hard for PPT adversaries to the hardness of the decisional H-LWE problem with noise-based semantic security construction).

The Data Encapsulation Mechanism (DEM) with AES-256-GCM gives AEAD (Authenticated Encryption with Associated Data) security as defined by NIST in SP 800-38D, where confidentiality is assured using counter mode encryption and integrity using Galois Field-based message authentication codes (GHASH). The nonce is generated according to this RFC, which guarantees that no two encrypted messages ever have the same prefix, and thus the counter cannot be reused in the lifetime of the cryptographic key, avoiding a nonce-reuse attack affecting confidentiality as well as integrity.

Decryption occurs only if the GHASH authentication tag verifies successfully; otherwise, any deviation in ciphertext, associated data, or tag results in verification failure, and plaintext is not accepted. These construction rules out chosen-ciphertext attacks, as adversary-controlled modifications can be detected, and normally, adaptive attack methods are not possible.

6.3. Side-Channel Resistance

The physical realization of cryptographic algorithms may result in information leakage over time, power traces, electromagnetic radiation, etc., and other side channels. HRA-HES has several defense strategies against these attacks. The compiled code ensures that all operations depending on the private key (polynomial multiplication, error sampling in the error distribution phase, and noise removal) run in fixed time, i.e., they always require the same number of CPU cycles and hence mitigate timing side channel attacks.

In the discrete Gaussian error sampler, we use a rejection sampling method with early termination, but only on rejections and never for acceptance; code paths always have a constant runtime. Blinding to certain intermediate values introduces random noise, resulting in decorrelation the power consumption from the sensitive secret of interest, which makes the power attack difficult. Informally speaking, masking splits sensitive values into random shares in such a way that by itself each share discloses no information;

Table 6. Side-Channel Attack categories and HRA-HES countermeasures

Attack Category	Complexity	Countermeasure	Effectiveness
Timing Attacks	Low	Constant-time implementation	High
Power Analysis (DPA)	Medium	Masking and blinding	High
EM Analysis (EMA)	Medium	EM shielding, noise injection	Med-High
Cache-Timing	High	Data-independent access patterns	Medium
Fault Injection	High	Error detection/correction codes	High
Template Attacks	High	Noise, shuffling	Med-High

Polynomial operations on masked shares are performed with a completely fresh randomness at every refresh point. Shuffling randomizes the ordering of independent operations, thereby breaking the linear correlation pattern that is used by an attacker in a differential power analysis. These countermeasures are then specifically applied to the KEM operations (KeyGen, Encaps, and Decaps) in which secret vectors are being manipulated—the AES-256-GCM DEM relies on hardware support for AES-NI, and hence we directly inherit side-channel resistance from constant-time microcode.

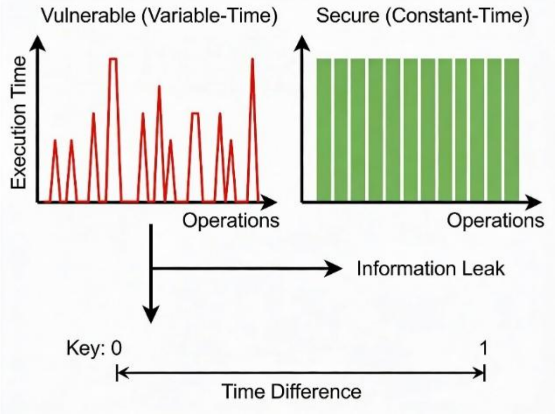

Fig 11. Timing Side-Channel Attack Concept – Comparison of vulnerable (variable-time) versus secure (constant-time) implementations showing how execution time must remain independent of secret data to prevent information leakage.

6.4. Quantum Security Analysis

6.4.1. Resistance to Shor's Algorithm

Shor's quantum period-finding algorithm provides exponential speed-up against the integer factoring and discrete logarithm problems by first constructing a single-valued [3], periodic function f: ℤ_N → G and then using the Quantum Fourier Transform (QFT) to find its periodic nature. The algorithm requires different quantum security superpositions over function evaluations and phase-space analysis to identify repeating patterns. In the hyper-LWE setting, the relation $b \in A \cdot s \oplus e$ cannot be directly instantiated as such a function because hyperaddition ⊕ produces a set of possible outputs rather than a unique value. Any approach to building a period-finding oracle must accommodate the non-determinism of several-valued operations, as these operations break the phase relations that underlie QFT-based period detection. Although not a proper reduction, this can be seen as a good indication that the hyperring algebraic framework of H- H-LWE inherently protects it from attacks like Shor's algorithm, since such results provide quantum security sidestepping lattice-based assumptions.

6.4.2 Resistance to Grover's Algorithm

Grover' algorithm gives quadratic improvement in searching an unstructured key when the key space is of size N, with O(√ N) quantum evaluations vs O(N) classical evaluations needed. For a block cipher with k-bit keys, the effective security against Grover's algorithm is only k/2 bits. HRA-HES uses AES-256-GCM, where the session key K is a uniformly random 256-bit value derived from the H-LWE KEM output via SHA3-256. Against Grover's algorithm, AES-256 provides an effective security level of 2^{128} quantum operations, equivalent to 128 bits of classical security, which is sufficient for current and foreseeable near-term threat models.

Quantum Security Against Grover's Algorithm:
Effective_quantum_security = k / 2 (bits) (5)
where k is the classical key length in bits.
For AES-256-GCM:
- Classical security: 256 bits
- Quantum security: 256 / 2 = 128 bits (still sufficient)

Table 7. Classical vs. Quantum security levels.

Component	Classical (bits)	Quantum (bits)	Resilience
H-LWE KEM (n=512)	128	128	High (conjectured)
AES-256-GCM	256	128	High (proven)
SHA3-256 (KDF)	256	128	High (proven)
Overall HRA-HES	256	128	High

The session key is further protected by the necessity of first breaking the H-LWE KEM to access encrypted key material; an adversary must simultaneously offer both the KEM and perform a key search against AES-256, a dual security requirement.

6.5. Security Against Lattice Reduction Attacks

Classical lattice reduction algorithms attempt to find short vectors in the lattice spanned by rows of the public matrix A, which would enable recovery of the secret vector s. The security of LWE-based schemes against lattice reduction depends on the gap between the error magnitude and the minimal lattice distance; when the error is sufficiently large relative to the shortest lattice vectors, reduction algorithms cannot distinguish signal from noise. H-LWE inherits this property: the discrete Gaussian error with σ = 3.2 is chosen such that the expected length of error vectors far exceeds the expected length of short lattice vectors for our parameter set [3].

The multi-valued nature of hyperaddition imposes extra algebraic complexity: lattice-reduction algorithms are naturally formulated for single-valued operations and do not directly handle the set-valued outputs of hyperaddition nor its non-deterministic algebraic behavior, which complicates their adaptation. This combinatorial complexity also increases the resistance against lattice-based classical attacks [5].

7. Conclusion and Future Work

HRA-HES is presented as a hyperring-based hybrid cryptosystem that aims to offer post-quantum security while remaining efficient and compatible with existing AES-based infrastructures. It formalizes the Hyperring Learning With Errors (H-LWE) problem over polynomial hyperrings, extending classical LWE to multi-valued algebraic structures and heuristically arguing that hyper addition disrupts the periodicity exploited by Shor-type quantum attacks. The scheme couples an H-LWE KEM with AES-256-GCM in a KEM–DEM composition to achieve authenticated bulk encryption with quantum-resilient key establishment.

On commodity hardware, HRA-HES attains about 850 Mbps encryption throughput, key-generation latency of 2.1 ms, and peak usage near 15% CPU and 45 MB memory, improving over classic RSA–AES baselines by up to ≈44% and remaining competitive with NIST-standardized schemes such as ML-KEM-768. The security analysis covers classical algebraic and side-channel attacks, as well as quantum threats based on Shor’s and Grover’s algorithms, and incorporates practical countermeasures (e.g., constant-time implementations).

Future work focuses on strengthening theory, implementations, and deployment. Formally relating H-LWE to standard LWE or Ring-LWE would solidify the hardness foundation. Hardware acceleration (FPGA/ASIC) could push throughput to multi-gigabit rates with lower energy per bit, while “lightweight” parameter sets targeting 64–96-bit security would address IoT and mobile constraints. Integration into real protocols (such as TLS-like handshakes) and progress toward IETF/IEEE/ISO-style profiles, together with systematic benchmarking against other post-quantum schemes, would clarify where hyperring-based hybrids provide the greatest practical advantage.

References

[1] W. Stallings, Cryptography and Network Security: Principles and Practice, 8th ed. Pearson, 2020.

[2] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in Proc. 35th Annu. Symp. Found. Comput. Sci., 1994, pp. 124–134.

[3] M. Mosca, "Cybersecurity in an era of quantum computers," IEEE Secur. Priv., vol. 16, no. 5, pp. 38–41, 2018.

[4] L. K. Grover, "A fast quantum mechanical algorithm for database search," in Proc. 28th Annu. ACM Symp. Theory Comput., 1996, pp. 212–219.

[5] D. Stebila and M. Mosca, "Post-quantum key exchange for the TLS protocol," in Proc. 31st Annu. Comput. Secur. Appl. Conf., 2016, pp. 1–15.

[6] National Institute of Standards and Technology (NIST), FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard, 2024.

[7] National Security Agency (NSA), "The commercial national security algorithm suite 2.0," Cybersecurity Advisory, 2022.

[8] J. Bos et al., "CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM," in 2018 IEEE Eur. Symp. Secur. Priv. (EuroS&P). IEEE, 2018, pp. 353–367.

[9] D. Stebila and M. Mosca, "Post-quantum key exchange for the TLS protocol," in Proc. 31st Annu. Comput. Secur. Appl. Conf., 2016, pp. 1–15.

[10] P. Corsini and V. Leoreanu, Applications of Hyperstructure Theory. Springer, 2003.

[11] B. Davvaz, Polygroup Theory and Related Systems. World Scientific, 2013.

[12] M. Akbiyik, "Hyperring-based coding theory and its applications," J. Algebra Appl., vol. 22, no. 4, 2023.

[13] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," J. ACM, vol. 56, no. 6, pp. 1–40, 2009.

[14] M. Dworkin, "Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC," NIST Special Publication 800-38D, 2007.

[15] Bellare, M., and Rogaway, P., “Robust Computational Secrecy and Authentication,” CCS ’93, 1993.

[16] Bellare, M., and Namprempre, C., “Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm,” ASIACRYPT 2000, LNCS 1976, 2000.

[17] Bernstein, D. J., Lange, T., and Niederhagen, R., “Post-quantum cryptography,” Nature, 549, 2017.

[18] Kaplan, M., Leurent, G., Leverrier, A., and Naya-Plasencia, M., “Quantum Differential and Linear Cryptanalysis,” CRYPTO 2016, LNCS 9814, 2016.

[19] Peikert, C., “A Decade of Lattice Cryptography,” Foundations and Trends in Theoretical Computer Science, 10(4), 2016.