Optimized HPPK Cryptography for Post-Quantum Security

Randy Kuang

Quantropi Inc. , Ottawa, Canada

Abstract. In this paper, we present an optimized construction of the Homomorphic Polynomial Public Key (HPPK) cryptosystem, a novel framework designed to provide enhanced security and efficiency in the post-quantum era. Our work introduces a layered cryptographic design that combines modular arithmetic permutations with an innovative additive random masking technique. This approach effectively obscures the underlying factorizable structure of the public key, thereby mitigating vulnerabilities to known lattice reduction attacks and other algebraic cryptanalyses. The security of our scheme is formally grounded in the computational hardness of three new problems: the Hidden Modulus Product Problem (HMPP), the HPPK Key Recovery Problem (HKRP), and the HPPK Secret Recovery Problem (HSRP). We demonstrate through rigorous analysis that the optimal attacks on our scheme are computationally infeasible for appropriately chosen parameters. Furthermore, we show that HPPK achieves remarkably compact key, ciphertext, and signature sizes, offering a significant advantage over leading NIST post-quantum finalists such as Kyber, Dilithium, and Falcon, particularly in bandwidth-constrained environments. The HPPK cryptosystem offers a compelling and mathematically-grounded solution for next-generation cryptography, delivering both provable security and practical efficiency.

Keywords: Cryptography · Post-Quantum Cryptography · PQC · Ken Encapsulation Mechanism · KEM · Digital Signature · Homomorphic Cryptography · Symmetric Encryption · Asymmetric Encryption

1 Introduction

The *Homomorphic Polynomial Public Key* (HPPK) cryptosystem, originally proposed by Kuang and Perepechaenko in 2023 [KP23], introduced a novel polynomial-based post-quantum key encapsulation mechanism (KEM). Its design centers on two private univariate polynomials $f(x)$, $h(x) \in \mathbb{F}_p[x]$ that generate product polynomials $p(x,u) = f(x)B(x,u)$ and $q(x,u) = h(x)B(x,u)$, where $B(x,u)$ is a shared multivariate polynomial. The system preserves the homomorphic ratio

$$\frac{f(x)}{h(x)} \equiv \frac{p(x,u)}{q(x,u)} \pmod{p},$$

with security originally based on hidden coprime pairs that permute coefficient vectors arithmetically. However, this structure has been shown to be susceptible to lattice reduction and algebraic attacks [JKP25].

The HPPK design was later extended to support a digital signature (DS) scheme [KPSL24], introducing signer anonymity and dual-ring encryption via the congruence $f(x)q(x,u) \equiv$

E-mail: randy.kuang@quantropi.com (Randy Kuang)

This work is licensed under a "CC BY 4.0" license.
Date of this document: 2025-08-12.

$h(x)p(x,u) \bmod p$. While the DS extension improved functionality, the underlying algebraic vulnerabilities remained.

In this work, we present a comprehensive optimization of HPPK to address these vulnerabilities and to enhance its security profile in the post-quantum context. Our approach introduces *permutation-based masking* techniques to obscure factorizable structures, tunable modulus-based parameters for adjustable security, and signature verification reorganizations that mitigate algebraic exposure. These refinements strengthen HPPK against known cryptanalytic vectors while preserving its original efficiency and design philosophy.

Contributions of this work:

- Introduction of random masking permutations in the KEM scheme to block lattice reduction and factorization attacks, with tunable security via modulus selection.
- Reorganization of constant terms in the DS scheme to reduce susceptibility to algebraic attacks during signature verification.
- A unified permutation-based design philosophy that strengthens both KEM and DS without reliance on external entropy mechanisms.

These optimizations significantly increase HPPK's resistance to both key recovery and signature forgery, providing a more robust polynomial-based candidate for post-quantum cryptography.

The remainder of this paper is organized as follows. Section 2 reviews related work in post-quantum cryptography. Section 3 details the optimized HPPK construction for both KEM and DS. Section 4 discusses attack surfaces and analyzes the impact of our enhancements. Finally, Section 5 concludes the paper.

2 Related Work

Public key cryptography has historically relied on the hardness of problems such as integer factorization, discrete logarithms, and elliptic curve discrete logarithms. Shor's quantum algorithm [Sho94, Sho99] breaks these assumptions, prompting the urgent development of post-quantum cryptography (PQC). The NIST PQC standardization effort [NIS21, NIS22, oST23a, oST23b, oST23c] has focused on lattice-based schemes (e.g., Kyber [ABD+20], Saber, NTRU), code-based McEliece, multivariate polynomial systems, isogeny-based protocols, and hash-based signatures (e.g., Dilithium [LDK+20], SPHINCS+ [ABB+20]).

While lattice-based schemes dominate PQC research, they are not without drawbacks, including large key sizes, implementation complexity, and susceptibility to side-channel attacks. This has motivated exploration of alternative algebraic frameworks, including polynomial-based systems, which can offer distinct trade-offs in terms of performance, key size, and security assumptions.

The original HPPK proposal [KP23] introduced a homomorphic ratio mechanism over finite fields, but subsequent analyses [JKP25] revealed structural weaknesses that could be exploited by lattice or algebraic cryptanalysis. Extensions to digital signatures [KPSL24] broadened functionality but did not fully resolve these weaknesses.

Our work builds upon these foundations, introducing targeted optimizations that address the identified vulnerabilities while preserving the desirable properties of HPPK. In doing so, we contribute to the broader effort of diversifying the post-quantum cryptographic landscape with schemes grounded in alternative mathematical structures.

3 Optimized HPPK Construction

This section presents the construction of our optimized Homomorphic Polynomial Public Key (HPPK) cryptosystem, which builds upon earlier work [KP23, KPSL24]. The design is rooted in two private univariate polynomials $f(x), h(x) \in \mathbb{F}_p[x]$ and a shared multivariate noise polynomial:

$$\begin{cases} f(x) = \sum_{i=0}^{\lambda} f_i x^i, \\ h(x) = \sum_{i=0}^{\lambda} h_i x^i, \\ B(x, u_1, \ldots, u_m) = \sum_{j=1}^{m} B_j(x) u_j, \quad B_j(x) = \sum_{i=0}^{n_b} b_{ij} x^i. \end{cases}$$

This multivariate structure enables both key encapsulation and digital signatures within a unified cryptographic framework. Our primary contribution is the introduction of mathematically-grounded optimizations designed to enhance the security of both the Key Encapsulation Mechanism (KEM) and Digital Signature (DS) schemes against known cryptanalytic attacks.

3.1 Key Encapsulation Mechanism (KEM)

3.1.1 Key Generation

The rational polynomial $\frac{f(x)}{h(x)}$ is encoded as:

$$\begin{aligned} \frac{f(x)}{h(x)} \pmod p &= \frac{f(x)B(x,u)}{h(x)B(x,u)} \pmod p \\ &= \frac{\vec{x}^T \cdot \mathbf{p} \cdot \vec{u}}{\vec{x}^T \cdot \mathbf{q} \cdot \vec{u}} \pmod p, \end{aligned} \tag{1}$$

where $\vec{x}^T = (1, x, \ldots, x^n)$ and n depends on the degrees of $f(x)$, $h(x)$, and $B_j(x)$. The matrices $\mathbf{p}, \mathbf{q} \in \mathbb{F}_p^{(n+1)\times m}$ have entries derived from the convolution of polynomial coefficients:

$$p_{ij} = \sum_{s+t=i} f_s b_{tj}, \quad q_{ij} = \sum_{s+t=i} h_s b_{tj}.$$

To protect these initially factorizable coefficients against lattice reduction attacks, two independent modular arithmetic permutations are applied via coprime pairs (R_1, S_1) and (R_2, S_2):

$$e = \frac{R_1 \cdot \square \pmod{S_1}}{R_2 \cdot \square \pmod{S_2}}, \quad e^{-1} = \frac{R_1^{-1} \cdot \square \pmod{S_1}}{R_2^{-1} \cdot \square \pmod{S_2}}, \tag{2}$$

with $\gcd(R_1, S_1) = \gcd(R_2, S_2) = 1$ and bit-lengths $L = \log_2 S_1 = \log_2 S_2 > 2 \log_2 p$. The symbol $\square$ denotes a generic placeholder for an input variable. This permutation operator is unitary, satisfying $e^{-1}e = 1$. Applying $e^{-1}e = 1$ to Eq. (1) yields the public key representation from the original HPPK design:

$$\frac{f(x)}{h(x)} \pmod p = \left(e^{-1} \frac{\vec{x}^T \cdot \mathbf{P} \cdot \vec{u}}{\vec{x}^T \cdot \mathbf{Q} \cdot \vec{u}}\right) \pmod p, \tag{3}$$

where the coefficients are:

$$P_{ij} = R_1 \cdot p_{ij} \pmod{S_1}, \quad Q_{ij} = R_2 \cdot q_{ij} \pmod{S_2}. \tag{4}$$

Since $\mathbf{p}$ and $\mathbf{q}$ possess an inherent factorizable structure, recent lattice reduction attacks (LRA) [JKP25] have shown that this structure can be exploited to recover private keys efficiently. To effectively mitigate this vulnerability, we introduce a random masking

layer—an additive permutation—designed to explicitly break this factorization without compromising the correctness of the scheme. This is achieved by enforcing the masking constraint:

$$\sum_{j=1}^{m} d_j u_j \equiv 0 \pmod{p}, \tag{5}$$

where $d_j \in \mathbb{F}_p$ are fixed, publicly known coefficients. Next, random masking coefficients $c_i^{(p)}, c_i^{(q)} \in \mathbb{F}_p$ are independently sampled to construct masking matrices:

$$\mathbf{M}_p = \{c_i^{(p)} d_j\}_{i,j}, \quad \mathbf{M}_q = \{c_i^{(q)} d_j\}_{i,j},$$

which in turn define the additive permutation operators:

$$\mathbf{A}_p(\square) = \mathbf{M}_p + \square \pmod{p}, \quad \mathbf{A}_q(\square) = \mathbf{M}_q + \square \pmod{p}.$$

Applying these additive permutations to the original factorizable coefficients yields the masked coefficients:

$$\begin{cases} p'_{ij} = \mathbf{A}_p(p_{ij}) = p_{ij} + c_i^{(p)} d_j \pmod{p}, \\ q'_{ij} = \mathbf{A}_q(q_{ij}) = q_{ij} + c_i^{(q)} d_j \pmod{p}. \end{cases}$$

Finally, to complete the public key formation, these masked coefficients are subjected to the modular multiplicative permutations via (R_1, S_1) and (R_2, S_2), resulting in the public key:

$$\mathbf{PK}_e = \begin{cases} P'_{ij} = R_1 \cdot p'_{ij} \pmod{S_1}, \\ Q'_{ij} = R_2 \cdot q'_{ij} \pmod{S_2}. \end{cases} \tag{6}$$

Because $c_i^{(p)}$ and $c_i^{(q)}$ are randomly chosen, the underlying factorizable polynomial structure is effectively masked, thereby fundamentally thwarting LRA attempts.

3.1.2 Encapsulation

Given the masked public key $\mathbf{PK}_e$ from Eq. (6), the encapsulation process proceeds as follows:

1. **Choose ephemeral secret.** Sample a secret $x \in \mathbb{F}_p$ and random noise variables $u_1, \ldots, u_{m-1} \in \mathbb{F}_p$. Compute the final noise variable u_m constrained by Eq. (5):

$$u_m = -d_m^{-1} \sum_{j=1}^{m-1} d_j u_j \pmod{p}.$$

2. **Compute encapsulation variables:** Calculate $x_{ij} = x^i u_j \pmod{p}$, for all $0 \leq i \leq n$, $1 \leq j \leq m$.

3. **Compute ciphertext components:** The ciphertext components c_p and c_q are computed by summing the products of public key elements and encapsulation variables:

$$c_p = \sum_{i=0}^{n} \sum_{j=1}^{m} P'_{ij} \cdot x_{ij}, \quad c_q = \sum_{i=0}^{n} \sum_{j=1}^{m} Q'_{ij} \cdot x_{ij}.$$

The resulting ciphertext is $C = (c_p, c_q)$.

3.1.3 Decapsulation

Given ciphertext $C = (c_p, c_q)$ and private key coprime pairs (R_1, S_1), (R_2, S_2), the decapsulation process recovers the shared secret:

$$k = \frac{R_1^{-1} \cdot c_p \pmod{S_1}}{R_2^{-1} \cdot c_q \pmod{S_2}} \pmod{p}.$$

The secret $x \in \mathbb{F}_p$ is then recovered by solving the polynomial equation $f(x) - kh(x) \equiv 0 \pmod{p}$. By design, the additive masking terms precisely vanish due to the constraint $\sum d_j u_j \equiv 0 \pmod{p}$, ensuring the correctness of the decapsulation while maintaining non-factorizability for LRA resistance during public key generation.

3.2 Digital Signature Scheme (DS)

3.2.1 Correctness

For the digital signature scheme, the fundamental expression in Eq. (1) is adapted to a product mode for each u_j as a set of equalities:

$$\vec{x}^T \cdot f(x)\mathbf{q}_j \pmod{p} = \vec{x}^T \cdot h(x)\mathbf{p}_j \pmod{p}, \tag{7}$$

where $\mathbf{q}_j$ and $\mathbf{p}_j$ represent the j-th column vectors of matrices $\mathbf{q}$ and $\mathbf{p}$, respectively. To mitigate potential factorization attacks arising from partial leakage of private key information, we strategically reorganize Eq. (7) by extracting and isolating the constant terms. This rearrangement ensures that the polynomials on both sides of the equation lack readily deterministic factorizations, thereby bolstering security. The reorganized equation is as follows:

$$s(x)b_{0j} + f(x)\sum_{i=1}^{n} q_{ij}x^i \pmod{p} = h(x)\sum_{i=1}^{n} p_{ij}x^i \pmod{p}, \tag{8}$$

with $s(x) = h_0 f(x) - f_0 h(x)$. Without accurate knowledge of their constant terms, the polynomials on both sides of Eq. (8) lose their deterministic factorizations. Now, let's introduce two permutation operators with two coprime pairs $\gcd(R_1, S_1) = 1$ and $\gcd(R_2, S_2) = 1$, with bit-length $L = 3l + 1$ for the purpose of Signature Encapsulation Mechanism (SEM):

$$e_1(\square) = R_1 \cdot \square \pmod{S_1}, \quad e_2(\square) = R_2^{-1} \cdot \square \pmod{S_2}.$$

and

$$e_1^{-1}(\square) = R_1^{-1} \cdot \square \pmod{S_1}, \quad e_2^{-1}(\square) = R_2 \cdot \square \pmod{S_2}.$$

so naturally, $e_1^{-1}e_1 = 1$ and $e_2^{-1}e_2 = 1$. Applying these operators to Eq. (8) yields the form for public verification:

$$\left[\left(SB_{0j} + F\sum_{i=1}^{n} Q_{ij}x^i\right) \bmod S_2\right] \bmod p = \left[\left(H\sum_{i=1}^{n} P_{ij}x^i\right) \bmod S_1\right] \bmod p, \tag{9}$$

with the public key components defined as:

$$\begin{cases} P_{ij} = R_1 \cdot p_{ij} \pmod{S_1} \\ Q_{ij} = R_2 \cdot q_{ij} \pmod{S_2} \\ B_{0j} = R_2 \cdot b_{0j} \pmod{S_2}. \end{cases} \tag{10}$$

and the signature elements (resulting from SEM encapsulation during signing):

$$\begin{cases} H = (R_1)^{-1}[\alpha h(x) \pmod p + r_h p] \pmod{S_1} \\ F = (R_2)^{-1}[\alpha f(x) \pmod p + r_f p] \pmod{S_2} \\ S = (R_2)^{-1}[\alpha s(x) \pmod p + r_s p] \pmod{S_2}. \end{cases} \tag{11}$$

Here, $\alpha, r_f, r_h, r_s \in \mathbb{F}_p$ are randomly chosen during the signing process. Equation (11) indeed represents a Signature Encapsulation Mechanism (SEM), analogous to the KEM. It is straightforward to observe that substituting Eq. (9) and Eq. (11) back into Eq. (9) correctly recovers Eq. (8). Equation (9) essentially acts as a signature verification equation that incorporates the private moduli S_1 and S_2. These moduli can be eliminated via the Barrett transformation algorithm:

$$a \cdot b \bmod n = ab - n \left\lfloor \frac{a\mu}{R} \right\rfloor, \tag{12}$$

where $\mu = \left\lfloor \frac{Rb}{n} \right\rfloor$ and $R = 4^L$ serves as the Barrett parameter. Applying this transformation and performing $\pmod p$ on both sides, the final verification equation becomes:

$$\Big(Sb'_{0j} - s_2 \Big\lfloor \frac{S\beta_{0j}}{R} \Big\rfloor\Big) + \sum_{i=1}^{n} \Big(Fq'_{ij} - s_2 \Big\lfloor \frac{F\nu_{ij}}{R} \Big\rfloor\Big) x^i \equiv \sum_{i=1}^{n} \Big(Hp'_{ij} - s_1 \Big\lfloor \frac{H\mu_{ij}}{R} \Big\rfloor\Big) x^i \pmod p, \tag{13}$$

with the public key matrices $\mathbf{PK}_v$ defined as:

$$\mathbf{PK}_v = \begin{cases} b'_{0j} = B_{0j} \pmod p, & \beta_{0j} = \left\lfloor \frac{RB_{0j}}{S_2} \right\rfloor \\ q'_{ij} = Q_{ij} \pmod p, & \nu_{ij} = \left\lfloor \frac{RQ_{ij}}{S_2} \right\rfloor \\ p'_{ij} = P_{ij} \pmod p, & \mu_{ij} = \left\lfloor \frac{RP_{ij}}{S_1} \right\rfloor \\ s_1 = S_1 \pmod p, & s_2 = S_2 \pmod p. \end{cases} \tag{14}$$

The secret key **SK** for the HPPK DS scheme is comprised of:

$$\mathbf{SK} = \begin{cases} b_{0j} \\ f(x), h(x) \\ (R_1, S_1), (R_2, S_2) \end{cases} \tag{15}$$

with the corresponding signature $sig = \{S, F, H\}$.

3.2.2 Signing

To generate a digital signature for a message m, the signer executes the following steps:

1. Compute a cryptographic hash of the message m using a NIST standardized hash algorithm: $x = \mathcal{H}(m) \in \mathbb{F}_p$.

2. Encapsulate $s(x), f(x)$ and $h(x)$ by applying a random scaling factor α and adding random multiples of p:

$$\begin{cases} \bar{s} = [\alpha s(x) \pmod p] + r_s \cdot p, \\ \bar{f} = [\alpha f(x) \pmod p] + r_f \cdot p, \\ \bar{h} = [\alpha h(x) \pmod p] + r_h \cdot p, \end{cases} \tag{16}$$

   with randomly chosen $\alpha, r_s, r_f, r_h \in \mathbb{F}_p$ as signature encapsulations or SEM.

3. Compute the final signature components using the private keys R_1, S_1, R_2, S_2:

$$\begin{cases} S = (R_2)^{-1} \cdot \bar{s} \pmod{S_2}, \\ F = (R_2)^{-1} \cdot \bar{f} \pmod{S_2}, \\ H = (R_1)^{-1} \cdot \bar{h} \pmod{S_1}. \end{cases}$$

4. The resulting digital signature for the message m is the tuple $\mathbf{Sig} = (S, F, H)$.

5. If $\mathcal{H}(m)$ has a longer bit-length than $\mathbb{F}_p$, a segmentation over the prime field is applied. The final signature would then be a concatenation of all signature segments, with each segment undergoing an independent encapsulation as per Eq. (16).

3.2.3 Verification

To verify a digital signature $\mathbf{Sig} = (S, F, H)$ for a given message m, the verifier performs the subsequent steps:

1. Compute the cryptographic hash of the message m using the same NIST standardized hash algorithm employed during the signing process: $x = \mathcal{H}(m) \in \mathbb{F}_p$.

2. Recompute polynomial coefficients using the provided signature **Sig** and the public key $\mathbf{PK}_v$:

$$\begin{cases} V_{0j} = \left(S \cdot b'_{0j} - s_2 \cdot \left\lfloor \frac{S\,\beta_{0j}}{R} \right\rfloor\right), \\ V_{ij} = \left(F \cdot q'_{ij} - s_2 \cdot \left\lfloor \frac{F\,\nu_{ij}}{R} \right\rfloor\right), \quad \text{for } i > 0 \\ U_{ij} = \left(H \cdot p'_{ij} - s_1 \cdot \left\lfloor \frac{H\,\mu_{ij}}{R} \right\rfloor\right). \end{cases}$$

   Then evaluate the left and right hand sides of Eq. (13):

$$V_j = \left(\sum_{i=0}^{n} V_{ij} x^i \right) \pmod{p},$$

$$U_j = \left(\sum_{i=1}^{n} U_{ij} x^i \right) \pmod{p}.$$

3. Verify the equality $V_j = U_j$ for all j based on the established security parameters of the scheme.

4. If the equality holds for all relevant j, the signature is deemed valid. Otherwise, the signature is considered invalid.

4 HPPK Cryptanalysis

This section formally introduces the computational hardness assumptions that underlie the security guarantees of the HPPK cryptosystem. We define three core cryptographic problems, each capturing a distinct security aspect of the scheme:

- **Hidden Modulus Product Problem (HMPP):** Underpins the security of the digital signature (DS) scheme by ensuring that an adversary cannot recover the hidden signing key—comprising the secret modulus and multiplier—from valid signatures and public information.

- **HPPK Key Recovery Problem (HKRP):** Captures the challenge of recovering the static private key components of the KEM, which we show is intractable due to a subproblem known as the HPPK Random Masking Problem.

- **HPPK Secret Recovery Problem (HSRP):** Quantifies the difficulty an adversary faces in recovering ephemeral secrets from observed ciphertext and the masked public key.

Each problem is introduced with a formal definition, followed by theorems stating their computational hardness and concise proof sketches. Detailed proofs and derivations are provided in Appendix A.

4.1 Hidden Modulus Product Problem (HMPP)

Definition 1 (Hidden Modulus Product Problem (HMPP))**.** Given random binary vectors $a_i \in \{0,1\}^l$, interpreted as integers, and public values A_i such that

$$A_i \equiv R \cdot a_i \pmod{S},$$

for some hidden modulus S and multiplier R, where $R \cdot a_i > S$, the problem is to recover the secrets S, R, and the original vectors a_i. Here, l is the bit-length of the binary vectors and L is the bit-length of the hidden modulus S.

Theorem 1 (Best Known Classical Attack on HMPP)**.** *The best known classical (non-quantum) attack on the HMPP requires approximately $\mathcal{O}(2^l \cdot L^4)$ operations, where l is the length of the binary vectors a_i and L is the bit-length of the hidden modulus S.*

Proof sketch: The attack proceeds by guessing candidate values for a_1 and constructing a lattice of small dimension. A BKZ-4 lattice reduction algorithm is applied to recover candidate values of S via the greatest common divisor (gcd) of short lattice vectors. Verification involves recomputing R and checking consistency with other A_i. No better quantum algorithms are currently known to substantially improve this complexity. Detailed steps appear in Appendix A.1.

4.2 HPPK Key Recovery Problem (HKRP)

The **HPPK Key Recovery Problem (HKRP)** captures the challenge of recovering the static private key components of the KEM. This attack proceeds in two stages, with the second stage formalized as the HPPK Random Masking Problem (HRMP).

Definition 2 (HPPK Random Masking Problem (HRMP))**.** Given masked coefficient matrices $\mathbf{P}' = (p'_{ij})$ and $\mathbf{Q}' = (q'_{ij})$ over $\mathbb{F}_p$, along with public coefficients d_j, the problem is to recover:

- the original private coefficient matrices $\mathbf{p} = (p_{ij})$ and $\mathbf{q} = (q_{ij})$,
- the secret random mask vectors $c^{(p)}$ and $c^{(q)}$,

subject to the linear masking constraint imposed by d_j.

Theorem 2 (Hardness of HRMP)**.** *Any adversary must perform an exhaustive deterministic search over at least $\mathcal{O}(p^{2(n+1)})$ possible mask vector pairs to solve the HRMP, where n is the polynomial degree and p is the prime modulus.*

Proof sketch: The adversary guesses the $2(n+1)$ unknown elements of $c^{(p)}$ and $c^{(q)}$, each in $\mathbb{F}_p$, yielding a search space of size $p^{2(n+1)}$. For each guess, the unmasked coefficients are computed as:

$$p_{ij} = p'_{ij} - c_i^{(p)} d_j, \quad q_{ij} = q'_{ij} - c_i^{(q)} d_j,$$

followed by polynomial factorization to verify whether the resulting polynomials share the common factor $B_j(x)$. Correct guesses pass this factorization test; incorrect guesses do not. Full details are in Appendix A.2.

Theorem 3 (Overall Complexity of HKRP)**.** *Recovering the HPPK KEM private key from the ephemeral public key PK_e proceeds in two stages:*

- Stage 1 (Feasible): *Use an HMPP-based lattice reduction attack to recover R_1, R_2 and deterministically compute the masked coefficients $\mathbf{P}'$ and $\mathbf{Q}'$. This requires a limited search over a small number of possible values, with complexity approximately $\mathcal{O}(2p \cdot \text{poly}(\log p))$.*
- Stage 2 (Intractable): *Exhaustively search for the mask vectors $c^{(p)}$, $c^{(q)}$ to solve the HRMP, with complexity $\mathcal{O}(p^{2(n+1)})$.*

Overall, the attack complexity is dominated by the exponential cost of Stage 2:

$$\mathcal{O}(p^{2(n+1)} \cdot \text{poly}(\log p)).$$

Proof sketch: Stage 1 involves a small number of lattice-based trials to recover secret multipliers. Stage 2 demands an exponential exhaustive search over mask vectors. Detailed proofs appear in Appendices A.1 and A.3.

4.3 HPPK Secret Recovery Problem (HSRP)

Definition 3 (HPPK Secret Recovery Problem (HSRP))**.** Given:

- masked public key coefficient matrices $\mathbf{P} = (P_{ij})$, $\mathbf{Q} = (Q_{ij})$ over $\mathbb{F}_p$,
- ciphertext components $c_p, c_q \in \mathbb{F}_p$,
- public masking constraint vector $d = (d_1, \ldots, d_m)$,

recover the ephemeral secrets:

- secret plaintext $x \in \mathbb{F}_p$,
- randomizer vector $u = (u_1, \ldots, u_m) \in \mathbb{F}_p^m$,

that satisfy the modular Diophantine system:

$$\begin{cases} \sum_{i,j} P_{ij} x^i u_j \equiv c_p \pmod{p}, \\ \sum_{i,j} Q_{ij} x^i u_j \equiv c_q \pmod{p}, \\ \sum_{j} d_j u_j \equiv 0 \pmod{p}. \end{cases}$$

Theorem 4 (Hardness of HSRP)**.** *Any classical or quantum adversary must perform at least $\mathcal{O}(p^{m-2})$ operations in the worst case to recover the plaintext x and randomizers u_j, where m is the number of randomizers.*

Proof sketch: The system has $m + 1$ unknowns and 3 modular equations, yielding roughly $m - 2$ degrees of freedom. An adversary must exhaustively guess over p^{m-2} candidates for the randomizers, then solve for remaining variables and verify consistency. No known algebraic or quantum shortcut reduces this complexity. See Appendix A.4 for details.

Remarks:

- HSRP corresponds to solving a modular Diophantine system involving both additive and multiplicative terms.
- The hardness stems from the exponential search space for randomizers.
- Typical parameters (e.g., $m = 4$) yield a complexity of $\mathcal{O}(p^2)$, ensuring strong practical security for sufficiently large p.

4.4 Security Summary

The security of the HPPK cryptosystem fundamentally relies on the computational hardness of the three problems above. Their relevance is summarized as follows:

- **For the KEM:** The adversary's optimal attack targets the HSRP, with complexity $\mathcal{O}(p^{m-2})$ per secret. For $m = 4$, this is $\mathcal{O}(p^2)$, which dominates over HKRP attacks. Thus, selecting two prime fields $\mathbb{F}_p$ for the masking rings optimally balances efficiency and security.
- **For the digital signature:** The preferred attack targets the HMPP, with complexity approximately $\mathcal{O}(2^l \cdot L^4)$. This remains significantly harder than direct key recovery from the public key, which requires $\mathcal{O}(2^L)$ operations [KPSL24].

Through layered additive masking, multiplicative encryption, and controlled randomization, HPPK forces adversaries to guess multiple independent secret components correctly, rendering both static key and ephemeral secret recovery computationally infeasible for appropriately chosen parameters.

Section 4.5 presents concrete key, ciphertext, and signature sizes informed by this security analysis.

4.5 HPPK Key, Ciphertext, and Signature Sizes

4.5.1 HPPK KEM Sizes and Complexity

The HPPK KEM private key (SK) consists of two linear polynomials, $f(x) = f_0 + f_1 x$ and $h(x) = h_0 + h_1 x$, and two secret multipliers R_1, R_2, totaling 6 elements in $\mathbb{F}_p$.

The public key PK_e includes two masked polynomial matrices P', Q', each of size $(n+1) \times m$. For typical parameters $n = 2$, $m = 4$, each matrix has 12 coefficients, for a total of 24 masked coefficients.

Ciphertexts contain two components (c_p, c_q) and are segmented to optimize size at various NIST security levels:

- Level I: 4 ciphertext segments
- Level III: 3 segments
- Level V: 2 segments

Larger p increases entropy per element, reducing required segments at higher security levels.

Choosing $m = 4$ ensures the hardness of the HSRP dominates security. For $m = 4$, best known attacks require $\mathcal{O}(p^2)$ operations, comparable to 128-bit security levels. Using two prime fields $\mathbb{F}_p$ optimizes efficiency and minimizes key and ciphertext sizes without reducing security.

Table 1: HPPK KEM sizes with optimized parameters $n = 2$, $m = 4$

NIST Level	p (bits)	|SK| (B)	|PK_e| (B)	|CT| (B)
I	64	48	192	64
III	96	72	288	72
V	128	96	384	64

Table 2 compares HPPK KEM sizes to NIST PQC finalists and candidates, illustrating HPPK's substantially more compact keys and ciphertexts, especially at bandwidth-constrained settings.

Table 2: HPPK KEM vs. NIST PQC KEMs

NIST Level	Algorithm	|SK| (B)	|PK| (B)	|CT| (B)
I	HPPK	**48**	**192**	**64**
	Kyber512	1632	800	768
	SABER	1568	672	736
	NTRU HRSS-701	1312	1147	1267
III	HPPK	**72**	**288**	**72**
	Kyber768	2400	1088	1088
	SABER	2352	992	1024
	NTRU HRSS-821	1568	1363	1523
V	HPPK	**96**	**384**	**64**
	Kyber1024	3104	1568	1568
	SABER	3136	1312	1344
	NTRU HRSS-1013	1824	1695	1895

4.5.2 HPPK Digital Signature Sizes and Parameters

The HPPK DS scheme relies on the hardness of the HMPP. The best known attack complexity is approximately $\mathcal{O}(2^l \cdot L^4)$.

Security parameters scale with NIST levels as:

$$l = \begin{cases} 64, & \text{Level I} \\ 96, & \text{Level III} \\ 128, & \text{Level V} \end{cases}, \quad \delta = 8, \quad L = 3l + 8,$$

and hash output length h (using SHA3) as:

$$h = \begin{cases} 256, & \text{SHA3-256} \\ 384, & \text{SHA3-384} \\ 512, & \text{SHA3-512} \end{cases}.$$

Each signature consists of three components (S, F, H), with total signature size:

$$\text{Sig} = 3 \cdot \left\lceil \frac{h}{l} \right\rceil \cdot \left\lceil \frac{L}{8} \right\rceil \text{ bytes.}$$

Table 3: HPPK DS key and signature sizes ($h = 256, 384, 512$ for Levels I, III, V)

NIST Level	l	L	$\mathcal{L}$	|SK| (B)	|PK$_v$| (B)	|Sig| (B)
I	64	200	400	132	300	300
III	96	296	592	196	444	444
V	128	392	784	260	588	588

Table 4 compares HPPK DS sizes with NIST PQC signature candidates, showing competitive compactness.

5 Conclusion and Future Work

In this work, we presented an optimized construction of the HPPK cryptosystem, a novel post-quantum framework built on rational polynomials. Our design introduces several

Table 4: HPPK DS vs. NIST PQC Signature Candidates

NIST Level	Algorithm	|SK| (B)	|PK| (B)	|Sig| (B)
I	HPPK	**132**	**300**	**300**
	Dilithium2	3504	1312	2420
	Falcon-512	1281	897	666
	Rainbow-Ia	4436	1693	1969
	GeMSS	4672	10432	2272
III	HPPK	**196**	**444**	**444**
	Dilithium3	5264	1952	3293
	Falcon-1024	2305	1793	1330
	Rainbow-IIIc	7636	2810	3299
	GeMSS	11200	24320	3648
V	HPPK	**260**	**588**	**588**
	Dilithium5	7568	2592	4595
	Falcon-1024	2305	1793	1330
	Rainbow-Vc	10652	3967	4861
	GeMSS	11200	24320	3648

key mathematical innovations to enhance its security and efficiency. By layering modular arithmetic permutations with a new random masking technique, we effectively obscure the underlying factorizable structure of the public key. This layered approach forces an adversary to solve two distinct and intractable computational problems—the HPPK Key Recovery Problem (HKRP) and the HPPK Secret Recovery Problem (HSRP) —to compromise the system.

The security of HPPK is formally grounded in these computational hardness assumptions, which we detailed in the cryptanalysis section. This security is achieved while maintaining remarkably compact key, ciphertext, and signature sizes. Our parameter choices, carefully aligned with NIST security levels, demonstrate that HPPK offers competitive or superior efficiency compared to leading post-quantum schemes such as Kyber, SABER, Dilithium, and Falcon. This advantage is especially pronounced in resource-constrained environments where bandwidth and computational efficiency are critical.

Future work will extend this research in several directions. First, comprehensive benchmarking across diverse hardware platforms is needed to validate HPPK's practical performance. Second, developing formal IND-CCA security proofs for the KEM and EUF-CMA proofs for the digital signature scheme will further solidify the theoretical underpinnings of HPPK. Finally, exploring additional mathematical optimizations to further improve performance and extend the architecture to support broader cryptographic protocols represents a promising avenue for continued development.

Overall, the HPPK cryptosystem offers a compelling and innovative solution for post-quantum security, delivering provable security, practical efficiency, and resilience through its layered mathematical design.

References

[ABB+20] Jean-Philippe Aumasson, Daniel J. Bernstein, Ward Beullens, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, and Bas Westerbaan. SPHINCS+: Specification document

(part of the submission package), 2020. URL: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf.

[ABD+20] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancréde Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Stehlö Damien. CRYSTALS-KYBER. Specification document (update from August 2021), 2020. URL: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf.

[JKP25] Daniel Johnson, Randy Kuang, and Daniel Panario. Cryptanalysis of hppk kem via lattice reduction and polynomial factorization, 2025. Prepare to submit it.

[KP23] Randy Kuang and Maria Perepechaenko. Homomorphic polynomial public key encapsulation over two hidden rings for quantum-safe key encapsulation. *Quantum Information Processing*, 22:315, 2023. doi:10.1007/s11128-023-04064-4.

[KPSL24] Randy Kuang, Maria Perepechaenko, Mahmoud Sayed, and Dafu Lou. Homomorphic polynomial public key with the barrett transformation for digital signature. *Academia Quantum*, 1, 2024. doi:10.20935/AcadQuant7353.

[LDK+20] V Lyubashevsky, L Ducas, E Kiltz, T Lepoint, P Schwabe, G Seiler, D Stehlé, and S Bai. CRYSTALS-Dilithium - Algorithm Specifications and Supporting Documentation (Version 3.1), 2020. URL: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf.

[NIS21] NIST. Status report on the second round of the nist post-quantum cryptography standardization process, July 2021. URL: https://csrc.nist.gov/publications/detail/nistir/8309/final.

[NIS22] NIST. Status report on the third round of the nist post-quantum cryptography standardization process, July 2022. URL: https://csrc.nist.gov/publications/detail/nistir/8413/final.

[oST23a] National Institute of Standards and Technology. Fips 203 (initial public draft): Module-lattice-based key-encapsulation mechanism standard. https://csrc.nist.gov/pubs/fips/203/ipd, 2023. ML-KEM (CRYSTALS-Kyber).

[oST23b] National Institute of Standards and Technology. Fips 204 (initial public draft): Module-lattice-based digital signature standard. https://csrc.nist.gov/pubs/fips/204/ipd, 2023. ML-DSA (CRYSTALS-Dilithium).

[oST23c] National Institute of Standards and Technology. Fips 205 (initial public draft): Stateless hash-based digital signature standard. https://csrc.nist.gov/pubs/fips/205/ipd, 2023. SLH-DSA (SPHINCS+).

[Sho94] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999. doi:10.1137/S0036144598347011.

Data Availability All data generated or analyzed during this study are included in this published article (and its Supplementary Information files).

Conflict of Interests The authors declare no conflicts of interest. All coauthors have thoroughly reviewed and endorse the manuscript's contents. We affirm that the submission represents original work and is not currently being reviewed by any other publication.

A Appendix: Detailed Proofs

A.1 Appendix: Hidden Modulus Product Problem (HMPP)

Problem Definition: Let λ be the security parameter. Define:

- $L = \text{poly}(\lambda)$: bit-length of the hidden modulus S,
- l: bit-length of the secret integers a_i,
- $t \geq 3$: number of public samples.

Sample uniformly:

$$S \xleftarrow{\$} [2^{L-1}, 2^L], \quad R \xleftarrow{\$} \mathbb{Z}_S^*, \quad a_i \xleftarrow{\$} \{0, \ldots, 2^l - 1\} \quad (i = 1, \ldots, t).$$

Compute public values:

$$A_i \equiv R \cdot a_i \pmod{S}, \quad \text{with } R \cdot a_i > S.$$

The challenge: recover the secrets R, S, and $\{a_i\}_{i=1}^t$.

Theorem 1 (restated). The best known classical attack has time complexity:

$$\mathcal{O}(2^l \cdot L^4),$$

where l is the bit-length of secret integers and L is the bit-length of the hidden modulus.

Proof. The attack proceeds in two phases.

Phase 1: Exhaustive Guessing and Lattice Reduction

1. For each candidate $\tilde{a}_1 \in \{0, \ldots, 2^l - 1\}$:
2. Use the relations (with unknown small integers k_{12}, k_{13}):

$$a_2 A_1 - a_1 A_2 = k_{12} S, \quad a_3 A_1 - a_1 A_3 = k_{13} S.$$

3. Build two small 4-dimensional lattice bases:

$$B = \begin{pmatrix} 2^l & 0 & A_1 & \tilde{a}_1 \\ 0 & 2^l & A_2 & -\tilde{a}_1 \\ 0 & 0 & 2^L & 0 \\ 0 & 0 & 0 & 2^L \end{pmatrix}, \quad B' = \begin{pmatrix} 2^l & 0 & A_1 & \tilde{a}_1 \\ 0 & 2^l & A_3 & -\tilde{a}_1 \\ 0 & 0 & 2^L & 0 \\ 0 & 0 & 0 & 2^L \end{pmatrix}.$$

4. Apply BKZ-4 lattice reduction to B and B' to obtain short vectors $\mathbf{b}_1$ and $\mathbf{b}'_1$.
5. Compute:

$$S_{\text{cand}} = \gcd(\mathbf{b}_1[3], \mathbf{b}'_1[3]).$$

If $S_{\text{cand}} \in [2^{L-1}, 2^L]$, proceed to Phase 2.

Phase 2: Verification and Secret Recovery

1. Compute:

$$R_{\text{cand}} \equiv A_1 \cdot \tilde{a}_1^{-1} \pmod{S_{\text{cand}}}.$$

2. Recover:

$$a_i^{\text{cand}} \equiv R_{\text{cand}}^{-1} \cdot A_i \pmod{S_{\text{cand}}}, \quad i = 2, \ldots, t.$$

3. Verify that all a_i^{cand} fall in $\{0, \ldots, 2^l - 1\}$. If so, output recovered $(\{a_i\}, R, S)$.

Correct vs. Incorrect Guess.

- If $\tilde{a}_1 = a_1$: the lattices B, B' truly contain short vectors corresponding to small multiples of S. BKZ-4 can efficiently find them, and the correct S is recovered.
- If $\tilde{a}_1 \neq a_1$: no such vectors exist; the gcd gives invalid S_{cand}.

Complexity.

- Number of guesses: 2^l.
- Per guess: two BKZ-4 reductions in 4 dimensions: $\mathcal{O}(L^4)$; verification steps are negligible.

Total: $\mathcal{O}(2^l \cdot L^4)$.

Summary. Even if a_i are random l-bit integers (not only binary), exhaustive guessing is still required, and the problem remains hard for large l. □

With appropriate parameters (e.g., $l = 64$ or 128), the best known attack on HMPP is computationally infeasible, supporting the security of HPPK cryptography.

The best known classical attack on the Hidden Modulus Product Problem requires an exhaustive search over 2^l guesses for a_1, with a polynomial-time lattice reduction and gcd verification step performed for each guess. Its overall complexity is $\mathcal{O}(2^l L^4)$, which grows exponentially with l. Consequently, for appropriately chosen practical security parameters where l is sufficiently large (e.g., $l \geq 64$ for Level I security), this problem is computationally infeasible.

□

A.2 Appendix: HPPK Random Masking Problem (HRMP)

Problem Definition: The HPPK Random Masking Problem (HRMP) asks: Given the masked coefficient matrices

$$p'_{ij} = p_{ij} + c_i^{(p)} d_j \pmod{p},$$

$$q'_{ij} = q_{ij} + c_i^{(q)} d_j \pmod{p},$$

where d_j are known public coefficients and p is the prime modulus, recover the private coefficient matrices $\mathbf{P} = (p_{ij})$, $\mathbf{Q} = (q_{ij})$, and the secret random mask vectors

$$c^{(p)} = (c_0^{(p)}, \ldots, c_n^{(p)}), \quad c^{(q)} = (c_0^{(q)}, \ldots, c_n^{(q)}).$$

Theorem 2 (restated). Any classical or quantum attacker must exhaustively search over a space of at least

$$\boxed{p^{2(n+1)}}$$

to solve HRMP.

Proof. The hardness derives from the number of unknowns and the absence of exploitable algebraic structure:

(1) Unknowns: The attacker must determine $2(n + 1)$ independent elements in the mask vectors $c^{(p)}$ and $c^{(q)}$, each in $\mathbb{F}_p$.

(2) Candidate space: Each element can take p possible values, so the total search space is

$$p^{n+1} \cdot p^{n+1} = p^{2(n+1)}.$$

(3) Reconstruction: For each candidate pair $(c^{(p)}, c^{(q)})$, the attacker computes:

$$p_{ij}^{\text{cand}} = p'_{ij} - c_i^{(p)} d_j \pmod{p},$$

$$q_{ij}^{\text{cand}} = q'_{ij} - c_i^{(q)} d_j \pmod{p}.$$

(4) Verification: The attacker verifies whether these reconstructed coefficients lead to polynomials sharing the a common factor $B_j(x)$ by polynomial factorization. This step is deterministic and feasible in polynomial time.

(5) Conclusion: Lacking algebraic shortcuts, the attacker must check all $p^{2(n+1)}$ possibilities. Hence, HRMP is deterministically hard with this complexity.

□

A.3 Appendix: HPPK Key Recovery Problem (HKRP)

Problem Definition: HKRP asks: Given the public keys

$$\mathbf{P}^{\text{pub}} = R_1 \cdot \mathbf{P}' \pmod{p}, \quad \mathbf{Q}^{\text{pub}} = R_2 \cdot \mathbf{Q}' \pmod{p},$$

recover the full private key: $\mathbf{P}$, $\mathbf{Q}$, mask vectors $c^{(p)}$, $c^{(q)}$, and multipliers $R_1, R_2 \in \mathbb{F}_p^*$.

Theorem 3 (restated). Recovering the full private key requires:

$$\boxed{\mathcal{O}(2p \cdot \text{poly}(\log p)) + \mathcal{O}(p^{2(n+1)})}$$

operations: first to deterministically recover R_1, R_2 and $\mathbf{P}'$, $\mathbf{Q}'$ (using HMPP, see Appendix A.1), then to exhaustively find $c^{(p)}, c^{(q)}$.

Proof. **(1) Stage 1: Recover R_1, R_2 and masked coefficients.** Using the HMPP approach, the attacker guesses one masked coefficient (e.g., p'_{11}). For each guess, runs a lattice reduction (complexity polynomial in $\log p$) to find R_1, and similarly for R_2. Total guesses: $2 \cdot p$. Hence, deterministic complexity: $\mathcal{O}(2p \cdot \text{poly}(\log p))$.

(2) Stage 2: Recover masks and private coefficients. After recovering $\mathbf{P}'$, $\mathbf{Q}'$, the attacker deterministically guesses mask vectors $c^{(p)}, c^{(q)}$ over $p^{2(n+1)}$ combinations. For each guess, computes:

$$p_{ij}^{\text{cand}} = p'_{ij} - c_i^{(p)} d_j, \quad q_{ij}^{\text{cand}} = q'_{ij} - c_i^{(q)} d_j,$$

and verifies whether these polynomials share the correct common factor. This verification is polynomial in n and $\log p$.

(3) Conclusion: Since both stages are deterministic, total attack complexity is dominated by the exponential step $\mathcal{O}(p^{2(n+1)})$.

□

Security note: Stage 1 is practical only if p is small; for realistic p (e.g., 128-bit), it becomes infeasible. The HRMP step remains deterministically hard due to the exponential search space.

A.4 Appendix: HPPK Secret Recovery Problem (HSRP)

Problem Definition: The HPPK Secret Recovery Problem (HSRP) is defined as follows: Given the public masked keys $(\mathbf{P}^{\text{pub}}, \mathbf{Q}^{\text{pub}})$, the ciphertext $C = (c_p, c_q)$, and the publicly known linear masking constraint $\sum_{j=1}^{m} d_j u_j \equiv 0 \pmod{p}$. The objective for an adversary is to recover the ephemeral secret plaintext $x \in \mathbb{F}_p$ and the randomizer vector $(u_1, \ldots, u_m) \in \mathbb{F}_p^m$ that were used during the encapsulation process.

Theorem 4 (restated). Let m be the number of randomizer variables used in the encapsulation. Even given the ciphertext (c_p, c_q) and the masked public key, any classical or quantum adversary must perform at least $\mathcal{O}(p^{m-2})$ operations to recover the tuple $(x, u_1, \ldots, u_{m-1})$, due to the public constraint reducing the randomizer space and the ciphertext providing two additional equations.

Proof. The proof analyzes the effective search space for the attacker given the information available:

(1) Initial Number of Unknowns: During encapsulation, the scheme uses m randomizers $u_1, \ldots, u_m \in \mathbb{F}_p$ and a secret plaintext $x \in \mathbb{F}_p$. This totals $m + 1$ independent unknown variables over $\mathbb{F}_p$ for each encapsulation.

(2) Reduction from Public Constraint: The scheme incorporates a public linear constraint on the randomizers:

$$\sum_{j=1}^{m} d_j u_j \equiv 0 \pmod{p}.$$

Assuming $d_m \neq 0$ (which can be ensured by reordering or design), this constraint allows one randomizer (e.g., u_m) to be expressed in terms of the others:

$$u_m = -d_m^{-1} \sum_{j=1}^{m-1} d_j u_j \pmod{p}.$$

This reduces the number of independent randomizer variables from m to $m - 1$. Consequently, the total number of independent unknowns the attacker needs to search over becomes x plus the $m - 1$ independent u_j's, yielding $1 + (m - 1) = m$ free variables to guess: $(x, u_1, \ldots, u_{m-1}) \in \mathbb{F}_p^m$. A naive brute-force search over this reduced space would require p^m operations.

(3) Reduction from Ciphertext Equations: The ciphertext components (c_p, c_q) provide two additional independent equations that relate the unknowns x and u_j to the public key components:

$$c_p = \sum_{i=0}^{n} \sum_{j=1}^{m} P'_{ij} \cdot (x^i u_j) \pmod{p},$$

$$c_q = \sum_{i=0}^{n} \sum_{j=1}^{m} Q'_{ij} \cdot (x^i u_j) \pmod{p}.$$

These two equations provide two constraints on the m remaining free variables $(x, u_1, \ldots, u_{m-1})$. Each equation can, in general, reduce the effective dimension of the solution space by approximately one.

(4) Effective Search Space and Non-Uniqueness: With m free variables and 2 effective equations from the ciphertext, the number of truly independent variables an attacker must exhaustively search over is reduced to $m-2$. This implies that, while there might be p^m candidate tuples initially, the existence of the two ciphertext equations (and the public masking constraint) means that the solution space for $(x, u_1, \ldots, u_{m-1})$ is of size approximately $\mathcal{O}(p^{m-2})$. This reflects the potential for non-uniqueness: multiple (x, u_j) tuples might map to the same ciphertext, but the attacker still needs to find *a* correct solution.

(5) Verification Cost per Candidate: For each candidate tuple $(x, u_1, \ldots, u_{m-1})$:

- Compute u_m using the public constraint: $u_m = -d_m^{-1} \sum_{j=1}^{m-1} d_j u_j \pmod{p}$. This is a constant number of operations.
- Calculate $x_{ij} = x^i u_j \pmod{p}$ for all i, j. This involves powers of x (up to n), multiplications, and additions over $\mathbb{F}_p$. This takes $\mathcal{O}(n \cdot m)$ field multiplications/additions.
- Compute the candidate ciphertext components c_p^{cand} and c_q^{cand} by summing over $(n+1)m$ terms: $c_p^{\text{cand}} = \sum_{i=0}^{n} \sum_{j=1}^{m} P'_{ij} x_{ij} \pmod{p}$, $c_q^{\text{cand}} = \sum_{i=0}^{n} \sum_{j=1}^{m} Q'_{ij} x_{ij} \pmod{p}$. This involves $\mathcal{O}(nm)$ field multiplications and additions.
- Finally, compare $(c_p^{\text{cand}}, c_q^{\text{cand}})$ with the actual ciphertext (c_p, c_q).

The total cost per verification step is dominated by matrix-vector multiplications and polynomial evaluations, which is $\mathcal{O}(nm)$ field operations. If field operations on L-bit numbers (where $L = \log_2 p$) take $\mathcal{O}(L^2)$ for standard multiplication (or $L^{\log_2 3}$ for Karatsuba), then the per-guess cost is $\mathcal{O}(nm \cdot L^2)$.

(6) Total Computational Complexity: The overall complexity of the attack is the product of the size of the reduced search space and the cost of verification per candidate:

$$\boxed{\mathcal{O}\big(nm \cdot (\log_2 p)^2 \cdot p^{m-2}\big)}.$$

The dominant exponential term is p^{m-2}, while $nm \cdot (\log_2 p)^2$ contributes as a polynomial factor (or $nm \cdot (\log_2 p)^{\log_2 3}$ for faster field arithmetic).

□

Summary: The HPPK Secret Recovery Problem (HSRP) poses a significant computational barrier to an adversary. Even with the public masking constraint reducing the number of free randomizers and the ciphertext providing additional equations, the remaining search space for the ephemeral secrets is still exponential in terms of p raised to the power of $(m-2)$. This ensures that HSRP remains computationally infeasible for appropriately chosen parameters p and m, thereby securing the confidentiality of individual encapsulated keys.