ACADEMIA QUANTUM

Review Article
Published: 2025-01-03
https://doi.org/10.20935/AcadQuant7457

Quantum permutation pad for quantum secure symmetric and asymmetric cryptography

Randy Kuang[1,*]

Academic Editors: Abdessatar Souissi, Steven K. Lamoreaux

Abstract

This review delves into the latest advancements in quantum-secure cryptography, focusing on the quantum permutation pad (QPP), a pivotal innovation proposed by Kuang et al. QPP harnesses the non-commutativity and generalized uncertainty derived from the Galois permutation group, making it highly suitable for cryptographic applications. The review underscores QPP's versatility across both symmetric and asymmetric cryptography through three core representations: matrix-based for classical encryption, quantum gates for quantum-native encryption, and arithmetic-based for multivariate public key systems such as Merkle–Hellman cryptosystems, multivariate public key cryptography (MPKC), and the most recent homomorphic polynomial public key (HPPK). In particular, QPP strengthens the security of HPPK's key encapsulation mechanism (KEM) and digital signature (DS) schemes, thus offering robust quantum resistance. This work further examines QPP's integration with various encryption techniques for enhancing resilience against quantum attacks. By addressing challenges in cryptographic complexity, key size optimization, and security enhancement, the review presents a thorough evaluation of QPP's role in fortifying cryptographic protocols for ensuring strong security foundations in the quantum computing era.

Keywords: *cryptography, post-quantum cryptography (PQC), quantum permutation pad (QPP), homomorphic polynomial public key (HPPK), knapsack, multivariate public key cryptography (MPKC), quantum key distribution (QKD), key encapsulation mechanism(KEM), digital signature (DS)*

Citation: Kuang R. Quantum permutation pad for quantum secure symmetric and asymmetric cryptography. *Academia Quantum* 2025;2. https://doi.org/10.20935/AcadQuant7457

1. Introduction

The realm of post-quantum cryptography (PQC) encompasses a diverse array of standardized schemes outlined by the National Institute of Standards and Technology (NIST). This overview succinctly encapsulates notable schemes, categorized according to their cryptographic underpinnings.

For key encapsulation mechanism (KEM), lattice-based contenders such as Kyber [1], BIKE [2], HQC [3], and code-based McEliece [4] take the spotlight. In addition, lattice-based Falcon [5], Dilithium [6], and hash-based SPHINCS$^+$ [7] emerge as prominent choices in the domain of digital signature (DS).

In a significant development in 2022, NIST announced standardized algorithms [8], endorsing Kyber for KEM and propelling McEliece, BIKE, and HQC into round 4. Simultaneously, NTRU [9] and Saber [10] were excluded from further consideration, while novel submissions for generic DS schemes were introduced [8]. NIST [11] recently announced its standardization decision with Kyber for KEM [1] and Dilithium [6] and SPHINCS$^+$ [7] for DS.

Lattice-based algorithms, exemplified by Kyber, BIKE, HQC, and Falcon, typically hinge on the short-vector problem (SVP) as the linchpin of their security. Code-based algorithms, as showcased by McEliece, derive security from the intricate decoding of random linear codes, providing robust post-quantum security. Hash-based algorithms, as exemplified by SPHINCS$^+$, are constructed based on the security of one-way trapdoors in hash functions. These nondeterministic polynomial time or NP-hard problems lay the groundwork for security against the looming threat of quantum computing.

Quantum key distribution (QKD) stands at the forefront of modern cryptographic protocols [12], leveraging quantum mechanics' foundational principles to establish secure communication in the face of emerging quantum computing threats. Unlike classical cryptography, which relies on complex mathematical problems [13–15], QKD exploits quantum properties, particularly photons. At its core, QKD uses quantum superposition and entanglement, allowing for secure cryptographic key exchange while promptly detecting any eavesdropping attempts. Its security is rooted in quantum indeterminacy, where measuring a quantum state disrupts it, thus revealing interception.

In the era of quantum computing threats to classical cryptography, QKD has emerged as a promising avenue for secure communication and has been proven to be theoretically secure [16, 17]. QKD can be classified into discrete-variable QKD (DV-QKD)

[1]Research, Quantropi Inc., Ottawa, ON K1Z 8P9, Canada.
[*]email: randy.kuang@quantropi.com

[18, 19] and continuous-variable QKD (CV-QKD) [20–22], both typically limited to transmission distances of about 100 km.

To extend the distance beyond this limit, Lucamarini et al. proposed twin-field QKD (TF-QKD) in 2018 [23]. Since then, various research efforts have successfully extended this distance to around 1000 km [24–31], marking a significant leap toward the practical implementation of long-distance quantum communication systems.

QKD fundamentally requires a shared public classical channel for post-processing, enabling both parties to establish a shared secret key in trusted environments. In untrusted scenarios, a pre-shared secret is necessary to prevent man-in-the-middle (MITM) attacks. Thus, all QKD schemes incorporate both post-processing and a pre-shared secret. Cryptographically, QKD functions as symmetric encryption, utilizing the uncertainty principle between 1-qubit computational and permutation bases for security. Kuang and Barbeau introduced photonic QKD implementations involving identity permutation gates (1-qubit computational basis) and XOR permutation gates (1-qubit permutation basis) within the Galois group over $\{0, 1\}$ [32], aligning QKD with Shannon's one-time pad (OTP) scheme [33]. Kuang and Bettenburg in 2020 [34] and Kuang and Barbeau in 2022 [32] proposed to extend OTP to quantum permutation pad or QPP for multi-qubit quantum symmetric encryption. The inherent non-commutativity of the Galois permutation group suggests that digital QKD offers more practicality than physical QKD, especially for multi-qubit systems [35]. Digital QKD offers an affordable, scalable solution for quantum-secure internet communication [34–36]. Kuang and Perepechaenko further demonstrated these QPP implementations in native quantum computing systems [37, 38], and the latest quantum circuit realizations are made by Burge and Mai and Barbeau in 2024 [39] and Chance in 2024 [40].

The QPP scheme is recognized as a quantum symmetric encryption method that utilizes a pre-shared secret permutation pad. Although permutation operators are generally non-commutative, some symmetric permutation operators are commutative and can be represented through mathematical operations such as modular exponentiation in RSA (Rivest–Shamir–Adleman) [13] and modular multiplication in the Merkle–Hellman cryptosystem or knapsack system [41]. The Merkle–Hellman system employs a single multivariate polynomial $y(x_1, \ldots, x_n)$ over a binary vector space, where its coefficients form a superincreasing sequence of integers. This sequence ensures that each integer is larger than the sum of all preceding integers, allowing for efficient decryption.

In the Merkle–Hellman cryptosystem, the encryption process involves using modular multiplication as a mathematical permutation operator. A coprime pair (M, W) serves as the encryption key, and the coefficients or the superincreasing sequence is encrypted using this modular multiplication operation. Specifically, each coefficient a_i is transformed into $a_i' = M \cdot a_i \mod W$, permuting the sequence within the ring $\{0, 1, \ldots, W-1\}$. The resulting public key, $\hat{p}a_i = M \cdot a_i \mod W = a_i'$, is used in conjunction with a binary-encoded plaintext $x_1, \ldots, x_n$ to generate the ciphertext.

During decryption, the recipient applies the reverse permutation operator $y = M^{-1} \cdot y' \mod W$, recovering the plain polynomial value. By leveraging the superincreasing property of the sequence, the recipient can efficiently solve the knapsack problem to extract the original plaintext. Although the Merkle–Hellman system was initially considered secure, it was broken by Shamir in 1982 [42].

Several variants of the knapsack system have since been proposed [43–46], but subsequent cryptanalysis has continued to reveal vulnerabilities [47–49].

To address the weaknesses of knapsack cryptosystems, Kuang and colleagues proposed a novel approach using two multivariate polynomials, $p(x, u_1, \ldots, u_m) \in \mathbb{F}_p$ and $q(x, u_1, \ldots, u_m) \in \mathbb{F}_p$, defined over two distinct vector spaces: one is the polynomial vector space for secret variables x, and the other is the linear vector space for noise variables $u_1, \ldots, u_m$ [50, 51]. This design replaces binary encoding with an integer in a prime finite field, enhancing security by incorporating noise variables within a key encapsulation mechanism (KEM). The relationship between these polynomials is captured by $\frac{p(x,u_1,\ldots,u_m)}{q(x,u_1,\ldots,u_m)} = \frac{f(x)}{h(x)} \mod p = k$, where k is derived from the ciphertext. Solving $f(x) - kh(x) = 0$ reveals the secret x, bypassing the vulnerabilities of superincreasing sequences and shifting from binary encoding to a secret integer in the prime field $\mathbb{F}_p$. The inclusion of noise variables enhances encapsulation through randomization, strengthening security.

To further enhance security, Kuang and Perepechaenko introduced two private coprime pairs, that is, (R_1, S_1) and (R_2, S_2), to encrypt the private coefficients of the polynomials $p(x, u_1, \ldots, u_m)$ and $q(x, u_1, \ldots, u_m)$, respectively. By leveraging the partial homomorphic property of modular multiplication, this design led to the creation of the homomorphic polynomial public key (HPPK) for a KEM [52]. The HPPK cryptosystem has since been extended to support DS schemes [53], expanding its applicability across various cryptographic protocols.

This review paper provides a detailed and systematical review of the QPP scheme, covering its representations and security features in Section 2. Section 3 offers a comprehensive review of QPP symmetric cryptography, discussing both classical implementations and quantum-native adaptations. Section 4 explores HPPK-based asymmetric cryptography, detailing its applications in KEM and DS, followed by a concluding discussion.

2. Quantum permutation pad

Quantum permutation gates, as a subset of quantum gates, exhibit unique properties that make them particularly suited for cryptographic applications. This section reviews the properties of these gates, their various representations across different cryptographic schemes, and the corresponding security implications. By exploring these aspects, we illustrate how quantum permutation gates provide a versatile foundation for both classical and quantum-secure encryption systems.

Quantum logic gates serve as the fundamental computing circuits enabling supercomputing power. Typically, these gates transform quantum states into superposition or entangled states, with the exception of a special class known as quantum permutation gates, which rearrange the computing basis. Examples include the single-qubit Pauli-X gate, the two-qubit CNOT or controlled-NOT gate, the SWAP gate, and the three-qubit Toffoli or CCNOT or controlled-CNOT gate.

For an n-qubit system, there exist $2^n!$ permutation gates $\{\hat{p}_i, i = 1, 2, \ldots, 2^n!\}$. This yields two permutation gates for a single-qubit system, 24 for a 2-qubit system, $40,320$ for a 3-qubit system, 2×10^{13} for a 4-qubit system, and over 10^{506} for an 8-qubit system. The entire space of permutation gates for an n-qubit system is

termed quantum permutation space (QPS) [32, 34] or quantum key space comparing with a classical key space of 2^n integers, a super exponential increase from 2^n to $2^n!$.

A typical permutation gate operation is denoted as follows:

$$\hat{p}_i|j\rangle = |k\rangle \tag{1}$$

where unique pairs $(j, k) \in [0, 2^n)$ exist for each permutation gate, amounting to 2^n such pairs. Consequently, a permutation gate operation establishes a bijective map from the set $\{0, 1, \ldots, 2^n-1\}$ to itself.

From its corresponding QPS, a set of n-qubit quantum permutation gates $\{\hat{p}_1, \ldots, \hat{p}_M\}$ can be selected to form a QPP [32, 34]. Utilizing a random shuffling algorithm like Wishart [54] to map a classical secret into a permutation gate, the Shannon entropy in bits of a random permutation gate e_i can be estimated as follows:

$$e_i = \log_2(2^n!) \approx (n - 0.42)2^n \tag{2}$$

for the i^{th} permutation gate when n is relatively large. Comparing this with the classical entropy in the Boolean algebra, where the key space spanned by $\{0, 1, \ldots, 2^n - 1\}$ attains a maximum Shannon entropy of n bits, the Shannon entropy of an n-qubit QPS experiences a super-exponential increase from n bits to $(n - 0.42)2^n$ bits.

Kuang and Bettenburg [34] and Kuang and Barbeau [32] established that QPP can be viewed as an extension of the OTP from the Boolean algebra in classical computing to linear algebra in quantum computing, with a 1-qubit QPP reverting to OTP.

2.1. Properties of quantum permutation gates

We summarize the basic properties of quantum permutation gates in a QPS as follows:

- **Bijectiveness**: For an n-qubit system, its computational basis is $\{|0\rangle, |1\rangle, \ldots, |2^n-1\rangle\}$. Each permutation gate $\hat{p}_i$ from the corresponding QPS uniquely rearranges the order of the computational basis. Then, there exists its reversed permutation gate $\hat{p}_i^\dagger$ which performs a reverse rearrangement back to its original computational basis: $\hat{p}_i^\dagger|k\rangle = \hat{p}_i^\dagger\hat{p}_i|j\rangle = |j\rangle$. Therefore, a permutation gate is unitary and reversible. This property aligns precisely with encryption and decryption through permutation and its reverse gate operations, respectively.
- **Composition of permutations**: Two permutations $\hat{p}_i\hat{p}_j$ compose a new permutation $\hat{p}_k = \hat{p}_i\hat{p}_j$ within the QPS. This implies that attempting to decrypt an encrypted state or cipher state largely performs a new encryption on the cipher state.
- **Non-commutativity**: Two permutation gates $\hat{p}_i$ and $\hat{p}_j$ generally meet $\hat{p}_i\hat{p}_j \neq \hat{p}_j\hat{p}_i$ or $[\hat{p}_i, \hat{p}_j] \neq 0$, implying that the order of permutation gates matters. It must be noticed that this property is true only when $n > 1$. In the quantum realm, the non-commutativity of permutation gates directly corresponds to the uncertainty principle in quantum mechanics and enables a randomly chosen QPP to be reusable without leaking the information of the QPP itself. This property makes the chosen ciphertext attack not directly applicable to QPP encryption [55], in contrast to the well-known OTP scheme.

These properties characterize the behavior of quantum permutation gates and underpin their utility in quantum computing and cryptography.

2.2. Representations of quantum permutation pad

QPP can be implemented using different representations tailored to specific cryptographic applications. These include matrix representations in classical computing systems, which have been widely explored for their efficiency in encryption schemes and post-quantum security applications [32, 34–36, 56]. In quantum native computing systems, QPP is often realized through gate representations, enabling direct implementation on quantum processors and facilitating quantum secure communications [37–40]. In addition, arithmetic representations of QPP are employed in public key cryptography, particularly in schemes such as HPPK and multivariate public key cryptography (MPKC), where they enhance both security and efficiency [50, 52]. This review focuses on these three primary representations, examining their roles in the development and implementation of cryptographic systems.

2.2.1. Matrix representations

For an n-qubit system, a quantum permutation gate is represented as a $\hat{p}_i[2^n \times 2^n]$ unitary and reversible matrix, commonly referred to as a permutation matrix. This matrix contains exactly one non-zero element per row and column, with all other entries being zero. The non-zero elements represent the permutation of basis states. The following is an example of a permutation matrix for a 3-qubit system:

$$\hat{p}_i = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Although there are only $2^3 = 8$ computational basis states in a 3-qubit system ($\{|0\rangle, |1\rangle, \ldots, |7\rangle\}$), the number of distinct 3-qubit permutation matrices is significantly larger, totaling $40,320$. These matrices provide the reversible and bijective transformations between basis states.

For example, consider the state vector $|3\rangle^T = (0, 0, 0, 1, 0, 0, 0, 0)$. Applying the permutation matrix $\hat{p}_i$ to this vector results in the following transformation:

$$\hat{p}_i|3\rangle = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = |1\rangle$$

In this case, $|1\rangle$ is the encrypted cipher state of $|3\rangle$ after applying the randomly chosen permutation matrix $\hat{p}_i$. Decryption can be performed using the transpose (or inverse) of the permutation matrix, $\hat{p}_i^\dagger$. Due to the unitary nature of $\hat{p}_i$, the decryption operation is simply the reverse transformation:

$$\hat{p}_i^\dagger|1\rangle = |3\rangle$$

This bijective property ensures that any valid encryption can be uniquely and accurately reversed.

To implement an n-qubit permutation matrix classically, the matrix requires $n \times 2^n$ bits of memory. However, the matrix can be compactly represented as a set of 2^n n-bit integers, where each element in the set corresponds to the position of the non-zero element in each row. Specifically, for a given row index i, the non-zero element in column j represents the value of the i^{th} element in the set:

$$\hat{p} \to S[2^n] = \{s_0, s_1, \ldots, s_i, \ldots, s_{2^n-1}\}$$

where s_i denotes the column index of the non-zero element in row i.

Similarly, the reverse permutation matrix $\hat{p}^\dagger$ can be stored as an additional set of $n \times 2^n$ bits, where the column index of each non-zero element corresponds to the index, and the row index corresponds to the value:

$$\hat{p}^\dagger \to T[2^n] = \{T[s_i] = t_i : i = 0, 1, \ldots, 2^n - 1\}$$

In this set, t_i is the row index of the non-zero element in column i.

With these representations, both encryption and decryption using permutation gates can be efficiently performed using classical computing resources. The operations can be expressed as follows:

$$\hat{p}|m\rangle = |T[m]\rangle = |C\rangle$$
$$\hat{p}^\dagger|C\rangle = |S[T[m]]\rangle = |m\rangle$$

Alternatively, the process can also be represented as follows:

$$\hat{p}|m\rangle = |S[m]\rangle = |C\rangle$$
$$\hat{p}^\dagger|C\rangle = |T[S[m]]\rangle = |m\rangle$$

Both approaches work due to the unitary and reversible nature of permutation gates. In practical applications, using 4- or 8-bit implementations for these operations often results in more efficient performance, as they align better with classical computing architectures.

2.2.2. Quantum gate representations

QPP can be implemented into quantum circuits to operate on quantum computing systems. A generic n-qubit permutation gate can be decomposed into 1- or 2-qubit gates [57, 58] for optimized performance. In 2024, Chance implemented QPP using Qiskit Runtime [40] and achieved a circuit depth of $\mathcal{O}(n)$ using only the SWAP gate. Burge, Mai, and Barbeau proposed their QPP circuit design in 2024 [39]. Liu et al. realized the implementation of n-qubit permutation gate with only CNOT gates in 2024 [59] and obtained the upper bond of $3(2^n - 1)$ CNOT gates required for an n-qubit permutation gate. Yu explored the quantum complexity of permutations in 2022 [60].

In 2022, Kuang and Perepechaenko, followed by Perepechaenko and Kuang in 2023, implemented QPP on a 5-qubit IBMQ system by inputting permutation matrices and using the Transpile compiler to generate permutation circuits. For 2-qubit permutation gates, the generated circuits had a depth of up to 15. However, the circuit depth increased exponentially, reaching over 100 for 4-qubit permutation gates. For instance, two 2-qubit permutation matrices

$$\hat{p}_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \hat{p}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

can be implemented in a 4-qubit circuit using Qiskit tools as shown in **Figure 1**. The transpiled circuits are shown in **Figure 2**.

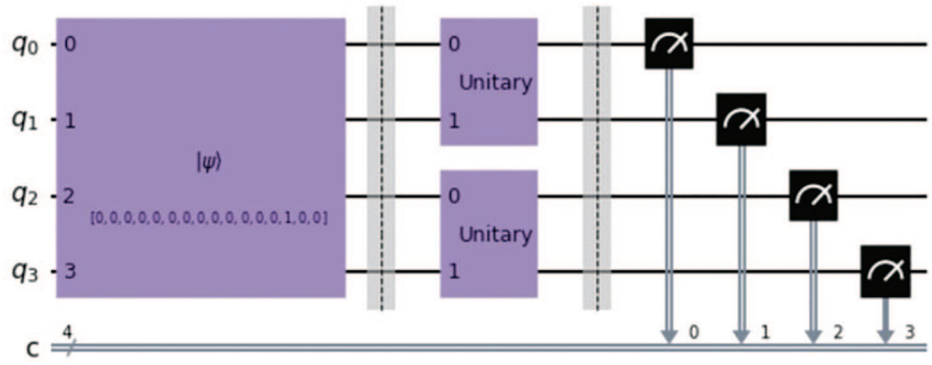

Figure 1 • Illustration of permutation gates using Qiskit tools. The leftmost section shows the initialization of 4 qubits. The top Unitary in the middle section represents the permutation gate $\hat{p}_1$, and the bottom Unitary represents the permutation gate $\hat{p}_2$. The rightmost section shows the measurement gates.

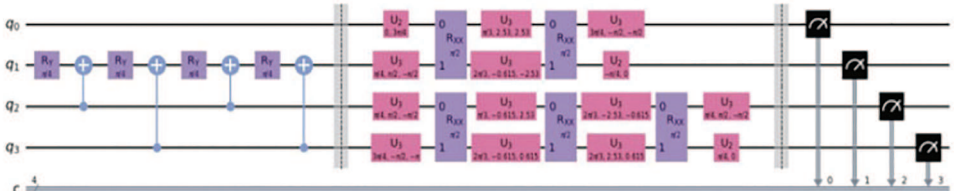

Figure 2 • Illustration of permutation circuits using the Qiskit Transpile compiler tool. The circuits for the two permutation gates are decomposed into two separate circuits: the top circuit for $\hat{p}_1$ and the bottom circuit for $\hat{p}_2$.

2.2.3. Arithmetic representations

In the realm of finite fields, arithmetic permutations play a pivotal role in cryptographic operations. These permutations, often expressed through modular arithmetic, contribute to the non-commutative and intricate transformations inherent in the Galois permutation group or symmetric group. This subsection delves into some critical arithmetic permutations that form the foundations of some cryptographic schemes.

XOR operation

In a binary finite field $GF(2^n)$, the XOR (eXclusive OR) operation (also known as bitwise addition modulo 2) is a fundamental arithmetic operation. XORing two bits results in a bit that is set if the two input bits are different and cleared if they are the same. For an n-bit QPS, there are a total of $2^n!$ permutations containing 2^n XOR permutations: $\hat{p}_i, i = 1, 2, \ldots, 2^n$. An XOR permutation gate operation on an n-bit message $|m\rangle$ can be simply written as follows:

$$\hat{p}_i|m\rangle = |(i-1) \oplus m\rangle = |c\rangle$$

This is the well-known OTP proven to be Shannon-perfect if the XOR permutation gate is randomly chosen and only used once because $|c\rangle = \hat{p}_c|0\rangle$ [33]. Let's use it to encrypt another n-bit message $|m'\rangle$,

$$\hat{p}_i|m'\rangle = |(i-1) \oplus m'\rangle = |c'\rangle$$

and then apply $\hat{p}_c$ on $|c'\rangle$

$$\begin{aligned} \hat{p}_c|c'\rangle &= |((i-1) \oplus m) \oplus ((i-1) \oplus m')\rangle \\ &= |((i-1) \oplus (i-1)) \oplus m \oplus m'\rangle = |m \oplus m'\rangle \end{aligned} \quad (3)$$

Here, the randomly chosen secret $i \in [0, 2^n - 1)$ is eliminated, leaving only the XOR of two low-entropy messages. If the random secret i is used only once, Eq. (3) becomes

$$\begin{aligned} \hat{p}_c|c'\rangle &= |\hat{p}_c \oplus c'\rangle = |((i-1) \oplus m) \oplus ((i' - 1) \oplus m')\rangle \\ &= |((i-1) \oplus (i' - 1)) \oplus (m \oplus m')\rangle \end{aligned} \quad (4)$$

Thus, the random secret cannot be eliminated.

Modular exponentiation

In the n-bit QPS, certain permutations can be represented by a modular exponentiation:

$$\hat{p} = \square^e \bmod S \quad (5)$$

where e is the exponent, S is the modulus, and $\square$ denotes an integer from the range $[0, S)$. In RSA public key cryptography [13], e and $S = pq$ (where p and q are prime numbers) form the public key. A corresponding private key d is chosen such that $ed \bmod \varphi(S) = 1$. Equation (5) acts as a permutation gate for encryption, while the reverse permutation for decryption is given by

$$\hat{p}^\dagger = \square^d \bmod S \quad (6)$$

Encryption of a secret message m is performed as follows:

$$\hat{p}|m\rangle = |m^e \bmod S\rangle = |c\rangle \in [0, S) \quad (7)$$

and decryption using the private key d is as follows:

$$\hat{p}^\dagger|c\rangle = |c^d \bmod S\rangle = |m^{ed \bmod \varphi(S)} \bmod S\rangle = |m\rangle \in [0, S) \quad (8)$$

Alternatively, some n-bit permutation gates in an n-bit QPS can be represented with another form of modular exponentiation:

$$\hat{p}(g) = g^\square \bmod q \quad (9)$$

where g is a generator and $\square$ denotes an integer from $[0, q)$. In the Diffie–Hellman key exchange algorithm [14], assuming secrets $a, b \in \mathbf{F}_q$, the process can be described as follows:

$$\begin{aligned} &\text{Alice:} \quad \hat{p}(g)|a\rangle = |g^a \bmod q\rangle = |A\rangle \rightarrow \text{Bob} \\ &\text{Bob:} \quad \hat{p}(g)|b\rangle = |g^b \bmod q\rangle = |B\rangle \rightarrow \text{Alice} \\ &\text{Alice:} \quad \hat{p}(B)|a\rangle = |B^a \bmod q\rangle = |g^{ab} \bmod q\rangle = |k\rangle \\ &\text{Bob:} \quad \hat{p}(A)|b\rangle = |A^b \bmod q\rangle = |g^{ab} \bmod q\rangle = |k\rangle \end{aligned} \quad (10)$$

Thus, both Alice and Bob establish their shared secret key k. Both RSA and Diffie–Hellman algorithms rely on the computational difficulty of the prime factorization problem and the discrete logarithm problem in classical computing. However, these problems are solvable using quantum computing via Shor's algorithm [61] because their moduli are publicly known.

Modular multiplication

For an n-bit QPS, certain permutations can be expressed as modular multiplication operations. Given an n-bit coprime pair (R_i, S_i) with $R_i < S_i$, a permutation gate can be written as follows:

$$\hat{p}_i = (R_i \cdot \square) \bmod S_i \quad (11)$$

where $\square$ is an integer in the range $[0, S_i)$. For an n-bit message $|m\rangle$, the gate $\hat{p}_i$ performs encryption as follows:

$$\hat{p}_i|m\rangle = |R_i \cdot m \bmod S_i\rangle = |c\rangle \quad (12)$$

Decryption is achieved with the reverse gate:

$$\hat{p}_i^\dagger|c\rangle = |R_i^{-1} \cdot R_i \cdot m \bmod S_i\rangle = |m\rangle \quad (13)$$

The matrix element $\langle j|\hat{p}_i|m\rangle$ is expressed as follows:

$$(\hat{p}_i)_{jm} = \langle j|\hat{p}_i|m\rangle = \langle j|R_i \cdot m \bmod S_i\rangle = \langle j|c\rangle = \delta_{jc} \quad (14)$$

Thus, each column of $\hat{p}_i$ contains a single non-zero element, showing that modular multiplication acts as a permutation operator.

Considering another arithmetic gate $\hat{p}_\ell = R_\ell \cdot \square \bmod S_\ell$, applying $\hat{p}_\ell$ and $\hat{p}_i$ in succession yields

$$\begin{aligned} &\hat{p}_\ell\hat{p}_i|m\rangle = \hat{p}_\ell|R_i \cdot m \bmod S_i\rangle = |R_\ell \cdot (R_i \cdot m \bmod S_i) \bmod S_\ell\rangle \\ &\hat{p}_i\hat{p}_\ell|m\rangle = \hat{p}_i|R_\ell \cdot m \bmod S_\ell\rangle = |R_i \cdot (R_\ell \cdot m \bmod S_\ell) \bmod S_i\rangle \\ &\qquad \rightarrow \hat{p}_\ell\hat{p}_i|m\rangle \neq \hat{p}_i\hat{p}_\ell|m\rangle \rightarrow \hat{p}_\ell\hat{p}_i \neq \hat{p}_i\hat{p}_\ell \end{aligned} \quad (15)$$

except for $S_i = S_\ell$. This non-commutativity, known as the generalized uncertainty principle, implies that for security purposes, the modulus S_i should ideally remain hidden. A secret or hidden modulus strengthens security by preventing an adversary from inferring critical system parameters, unlike in RSA or Diffie–Hellman. There exist approximately $\mathcal{O}(2^{2n})$ pairs of arithmetic permutation gates in an n-bit QPS [52].

Table 1 • Illustration of quantum permutation space as a function of n-qubit system, together with the secret length required to choose a permutation matrix, the entropy of quantum permutation space, and the number of permutation matrices required to achieve 256 bits of entropy

n (bits)	QPS dimension	Secret (bits)/permutation	Entropy (bits)	QPP size
2	24	8	4.5	57
3	40,320	24	15.3	17
4	2.09×10^{13}	64	44.25	6
6	1.28×10^{89}	384	296	1
8	$> 10^{506}$	2,048	1,684	1

The arithmetic permutation encryption in Eq. (11) used for public cryptography can be found in the earliest public key scheme called the Merkle–Hellman cryptography with the knapsack cryptosystem [41], by combining a superincreasing set of coefficients with a multivariate binary vector space for secret message encoding. The Merkle–Hellman cryptography had two weaknesses: the superincreasing set as coefficients in a plain multivariate polynomial and binary vector space for message encoding and decoding, which led to vulnerability to various attacks [49, 62]. By eliminating these weaknesses, Kuang, Perepechaenko, and Barbeau proposed the multivariate polynomial public key (MPPK) KEM in 2022 [50], and Kuang and Perepechaenko later upgraded MPPK KEM to the HPPK for key encapsulation and DS schemes in 2023 [52, 53] with a QPP of two arithmetic permutation gates to overcome the weaknesses of the Merkle–Hellman cryptography.

2.3. Quantum permutation pad security

The security of a QPP is influenced by its underlying representations, such as mathematically unstructured matrices and structured arithmetic, each offering unique security features. These representations contribute to the overall robustness of QPP against both classical and quantum adversaries. In the following subsections, we will provide a concise overview of QPP's security attributes, examining how these representations enhance its resistance to attacks and contribute to its suitability for quantum-secure cryptographic applications.

2.3.1. Matrix representations

A classical random bit string can be mapped to a permutation matrix through the Wishart algorithm [54]. For sufficient shuffling, the bit length of a random bit string is required to be $n \times 2^n$ bits long for a permutation matrix, but with an effective entropy $e = \log_2(2^n!) \approx (n - 0.42)2^n$ for relatively large n. For example, a 3-bit permutation matrix requires a secret of $3 \times 2^3 = 24$ bits. However, the equivalent entropy of a randomly selected 3-bit permutation matrix equals $\log_2(2^3!) = \log_2 40320 \approx 15.3$ bits. If a QPP achieves 256 bits of entropy with 3-bit permutation matrices, we need to select 17 permutation matrices.

Table 1 illustrates five QPSs with $n = 2, 3, 4, 6, 8$, together with the secret length to randomly choose one permutation matrix, equivalent Shannon entropy, and the number of permutation matrices required to achieve at least 256 bits of entropy.

It is clearly seen from **Table 1** that the size of QPS increases super-exponentially from 24 permutations at $n = 2$ to more than 10^{506} at $n = 8$, and the entropy of a QPS increases exponentially from 4.5 bits at $n = 2$ to 1684 bits at $n = 8$. For secure communication, both peers need to share a QPP for encryption and a QPP† for decryption. Assuming there are M permutation matrices per QPP, we require a memory space of $2 \times M \times n \times 2^n$ bits. Considering performance, it is wise to choose $n = 4$ or $n = 8$ to take advantage of computing architecture.

The total effective entropy for a pad with M permutation matrices is

$$e = \log_2(2^n!)^M \approx M \log_2(2^n!) \approx M(n - 0.42)2^n$$

bits for relatively large n. In digital QKD implementation [35], QPP is generated with $n = 8$ and $M = 64$, which offers a total of $64 \times 1,684 > 100,000$ bits of entropy. In this case, the total memory required for QPP and QPP† is 32 KB.

Due to the non-commutativity of permutation matrices, QPP encryption naturally prevents chosen ciphertext attacks [32, 55]. However, its bijectivity in permutation encryption not only transforms the statistical patterns in the plaintext into ciphertext but also makes it vulnerable to chosen plaintext attacks [55]. To avoid the chosen plaintext attack, some pre-processing techniques are recommended by Kuang and Bettenburg [34] and Kuang and Barbeau [32]. Section 3, or QPP symmetric cryptography, will discuss how to avoid these weaknesses and enable QPP security equivalent to a brute-force search complexity of $\mathcal{O}(M \log_2(2^n!))$.

2.3.2. Arithmetic representation

For modular multiplication permutation, n-bit arithmetic permutations form a subspace of the n-bit QPS, resulting in a significant reduction in entropy from $\log_2(2^n!) \approx (n - 0.42)2^n$ to approximately $\log_2 2^{2n} = 2n$ bits. Therefore, a QPP with M permutations offers $2nM$ bits of entropy. However, an arithmetic QPP is highly useful for both symmetric and asymmetric encryptions.

In the symmetric case with modular multiplication permutations, the communication peers share the same arithmetic QPP for encryption and decryption. For instance, the equivalent security of AES-256 can be achieved with either $n = 9$ and $M = 15$ or $n = 17$ and $M = 8$.

In the asymmetric case, an arithmetic QPP can be self-shared: the same QPP is used to encrypt the "plain public" key during key generation, producing a cipher public key, then to decrypt the received ciphertext and extract the secret. The simplest pad, with a single permutation gate, has been used in the Merkle–Hellman cryptosystem [41]. Therefore, the arithmetic QPP using for public key scheme may further be reduced in its security. We will discuss this in Section 4.5.4.

3. Quantum permutation pad symmetric cryptography

In Section 2.3, we noted that using a pure QPP for encryption can be vulnerable to chosen plaintext attacks and may leave statistical fingerprints in the ciphertext. To address these vulnerabilities and enhance security, Kuang and Barbeau introduced pre-randomization and random dispatching techniques [32]. These techniques strengthen QPP's role in symmetric cryptography by ensuring that the ciphertext remains resistant to statistical analysis and quantum attacks. **Figures 3** and **4** illustrate how QPP symmetric cryptography integrates pre-randomization and random dispatch, providing a more robust encryption framework. In this section, we will review different implementations of QPP in symmetric cryptography, including classical methods, pseudo-random number generation, digital QKD, and quantum computing, highlighting their contributions to quantum-secure encryption.

Given QPP's bijective mapping property, it can transform hidden structures in plaintexts into ciphertexts. To address this potential weakness, a common strategy involves pre-randomizing the plaintext to enhance both confusion and diffusion capabilities [32]. In **Figure 3**, a cryptographic pseudo-random number generator (PRNG) is seeded with the hash code of the shared secret key. The generated n-bit pseudo-random number x is XORed with an n-bit segment m from the plaintext input, producing an n-bit intermediate pseudo-random cipher m'. This pre-randomization mimics Shannon's OTP encryption, eliminating statistical fingerprints in the plaintext. The chosen PRNG must ensure both

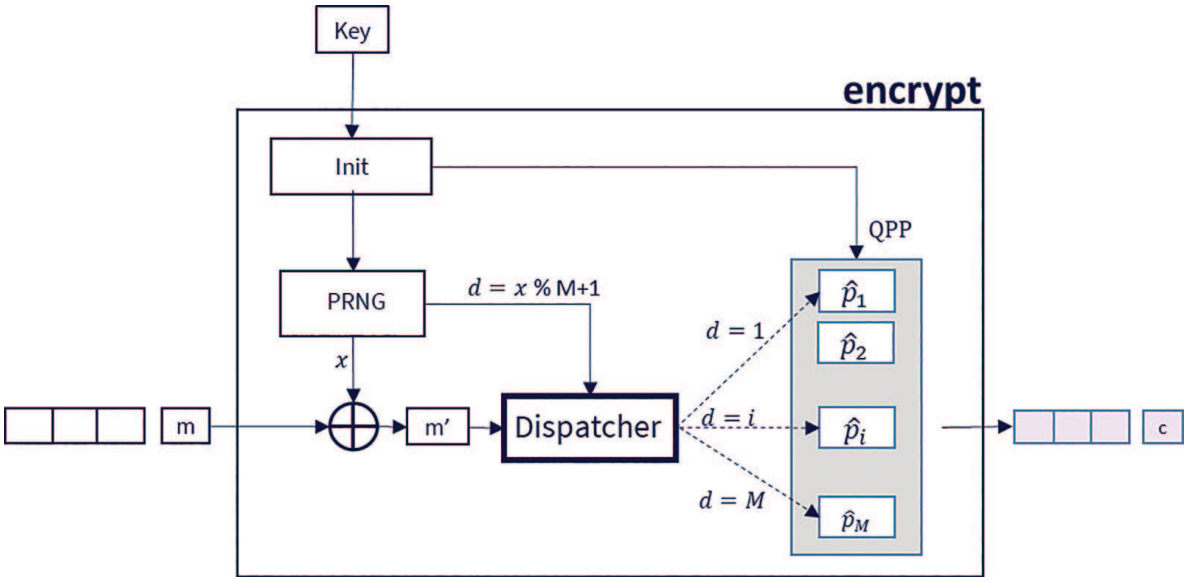

Figure 3 • QPP symmetric encryption involves **Init**, **PRNG**, **Dispatcher**, and **QPP** modules. **Init** takes the shared secret key bit string and generates an n-bit **QPP** pad using the Fisher–Yates shuffling algorithm. **Init** then hashes the secret key to seed a cryptographic PRNG generator. Input plaintext is segmented into n-bit segments, XORed with n-bit pseudo-random x (pre-randomization), and then dispatched to the d^{th} permutation matrix for encryption, resulting in the ciphertext.

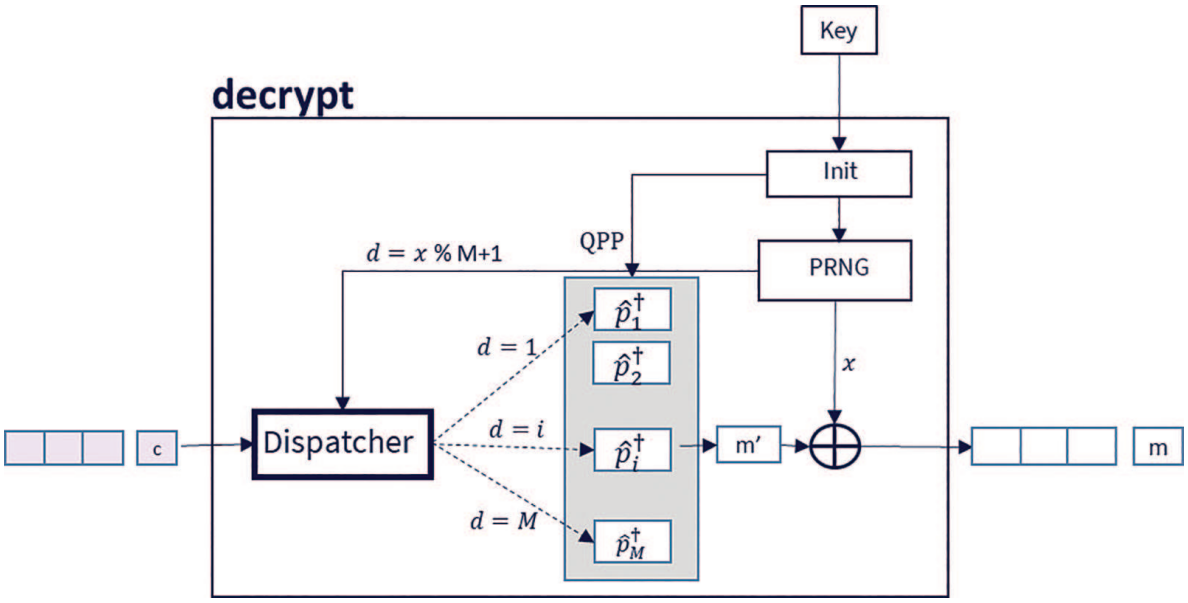

Figure 4 • QPP symmetric decryption involves the same modules as in **Figure S4**. The **Dispatcher** first dispatches each n-bit ciphertext to the correct permutation matrix. **QPP** then decrypts them, followed by post-randomization to output the original plaintext.

pseudo-randomness and high performance, with xoroshiro128+ being a typical PRNG for this purpose.

The **Dispatcher** in **Figure 3** sends the intermediate cipher m' to the d^{th} permutation gate $\hat{p}_d$ in QPP for encryption:

$$\hat{p}_d|m'\rangle = |c\rangle \tag{16}$$

The QPP outputs the final ciphertext c. This dispatching largely randomizes the relationship between permutation gates and cipher segments, preventing chosen plaintext attacks. For any intercepted n-bit cipher c, the permutation gate is chosen based on the pseudo-random number x, so a given c could be encrypted by any permutation gate from the QPP.

Pre-randomization and random dispatching significantly increase confusion and diffusion capabilities, creating an uncertainty barrier between the plaintext space and the QPP, as well as the ciphertext space, thereby enhancing the security of QPP symmetric cryptography against chosen plaintext attacks.

Figure 4 illustrates the decryption process of QPP symmetric cryptography. The ciphertext is segmented into n-bit segments and dispatched to their corresponding permutation gates based on the pseudo-random number from the PRNG. The decryption QPP must be transposed because $\hat{p}_d^\dagger \hat{p}_d = 1$. Then, the output from the QPP

$$\hat{p}_d^\dagger|c\rangle = \hat{p}_d^\dagger \hat{p}_d|m'\rangle = |m'\rangle \tag{17}$$

is XORed with the pseudo-random number x to retrieve the original plaintext m.

3.1. Classical implementations

QPP symmetric cryptography, as typically depicted in **Figures 3** and **4**, can be implemented in classical systems [56]. This implementation requires a minimal footprint of about 3KB for an 8-bit QPP, containing 1684M bits of entropy. Ciphertexts generated from QPP symmetric encryption exhibit excellent randomness, successfully passing all major randomness testing tools, including NIST, DieHarder, and ENT. Performance-wise, it operates similarly to a single round of AES encryption, achieving speeds over 10 times faster than AES-256.

3.2. Pseudo quantum random number

Due to the substantial entropy held by a QPP, QPP symmetric cryptography can also be used to develop a pseudo-quantum random number generator (pQRNG) [36]. Prominent PRNGs, such as Xorshift, invented by Marsaglia in 2003 [63] and its variants created by Vigna in 2016 [64] and Blackman and Vigna in 2022 [65] typically have a maximum seed size of 1,024 bits, as seen in the xoshiro1024 generator. In contrast, the pQRNG based on QPP cryptography employs 64 8-bit permutation gates, seeded with a 16KB binary string, which is the largest known seed size for any PRNG. The pQRNG boasts a theoretical internal state space of 2^{131072}. **Figure 5** illustrates a typical pQRNG implementation using QPP cryptography [36]. This pQRNG also includes an internal counter XORed with the generated pseudo-random number c, similar to QPP encryption. The counter can alternatively replace the internal PRNG x. The output pseudo-random numbers pass all standard randomness testing tools such as NIST, DieHarder, and ENT [36]. When the seed is replaced with a system random source, the pQRNG can produce non-deterministic pseudo-random numbers.

Furthermore, QPP cryptography can function as a quantum whitening algorithm for classical PRNGs and hardware/quantum random number generators (HRNGs or QRNGs) [36]. Physical RNGs generally produce biased random number such as found in popular QRNGs from idQuantique [66]. Quantum whitening is achieved by initializing QPP with the system random and using the output from a hardware RNG as the plaintext input. The resulting QPP ciphertext then exhibits excellent randomness suitable for cryptographic keys.

3.3. Digital quantum key distribution

Kuang and Barbeau [32] proved that QKD is equivalent to 1-qubit QPP with post-shared secret for measurement bases. The post-sharing is established based on the measurement bases at the receiving synchronized with the encoding bases at the transmission. In fact, QKD generally encodes classical key bits with two conjugate bases: $\{|0\rangle, |1\rangle\}$ and $\{|+\rangle, |-\rangle\}$. These two encoding bases are nothing else but the eigenbasis or computational basis of the identity gate $\hat{p}_0 = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and permutation basis of the NOT gate $\hat{p}_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. The quantum implementation of single qubit QPP would be the well-known QKD by leveraging the security from the uncertainty principle with a tradeoff of classical channel together with a pre-shared secret for post-processing, and the classical implementation of single qubit (bit) QPP is the well-known OTP with the limitation of single-time usage of the pre-shared key.

The implementation of multi-qubit QKD is technically challenging even for 2-qubit QKD, but the classical implementation of n-bit QPP is relatively simpler by using the matrix representation of permutation gates. Lou et al. implemented the so-called digital QKD or D-QKD and benchmarked by Deutsch Telekom in 2022 [35]. D-QKD leveraged the QPP symmetric cryptography and integrated with ETSI-014 standard interface for plug-and-play and interchangeable with physical QKD boxes [67], illustrated in **Figure 6**. The D-QKD cloud hosts a QRNG farm consisting of different QRNG generators from Quintessencelabs qStreamer [68] and idQuantique QRNG [69]. The communication for each connection segment is encrypted with QPP symmetric cryptography as shown in **Figures 3** and **4** with a QPP of 64 8-bit permutation matrices or equivalent to 100 kilobits of entropy. D-QKD demonstrated the key rate of 100s Mbps over internet connections cross continentals [35].

3.4. Quantum computing implementation

As discussed in Section 2.2.2, a QPP can be realized as a quantum circuit within a quantum computing system. This implementation involves using a compiler to decompose permutation gates into circuits composed of one- or two-qubit gates. QPP symmetric cryptography can thus be fully executed in a quantum computing environment, incorporating pre-randomization and random dispatching as additional quantum gates [37, 38].

Pre-randomization, which is essentially classical XORing, can be implemented using quantum CONTROL-NOT (CNOT) gates. Random dispatching can be realized with Control-Permutation gates, as illustrated in **Figures 7** and **8**. Due to constraints on quantum resources, the implementation of QPP cryptography on

IBM Quantum (IBMQ) systems is currently demonstrated with 2- or 3-qubit permutation gates, suitable for the available 5-qubit noisy IBM Quantum computers.

To achieve security comparable to AES-256, a 2-qubit QPP implementation requires 57 permutation gates chosen randomly from the 2-qubit QPS (which has a total of 24 gates). The resulting cipher qubits can be measured and recorded based on the most probable outcome. This prototype implementation has surprisingly demonstrated the equivalence between quantum and classical implementations. It also paves the way for future hybrid quantum-secure communications, bridging classical and quantum computing systems, thereby positioning QPP as a bridge from classical to quantum communications.

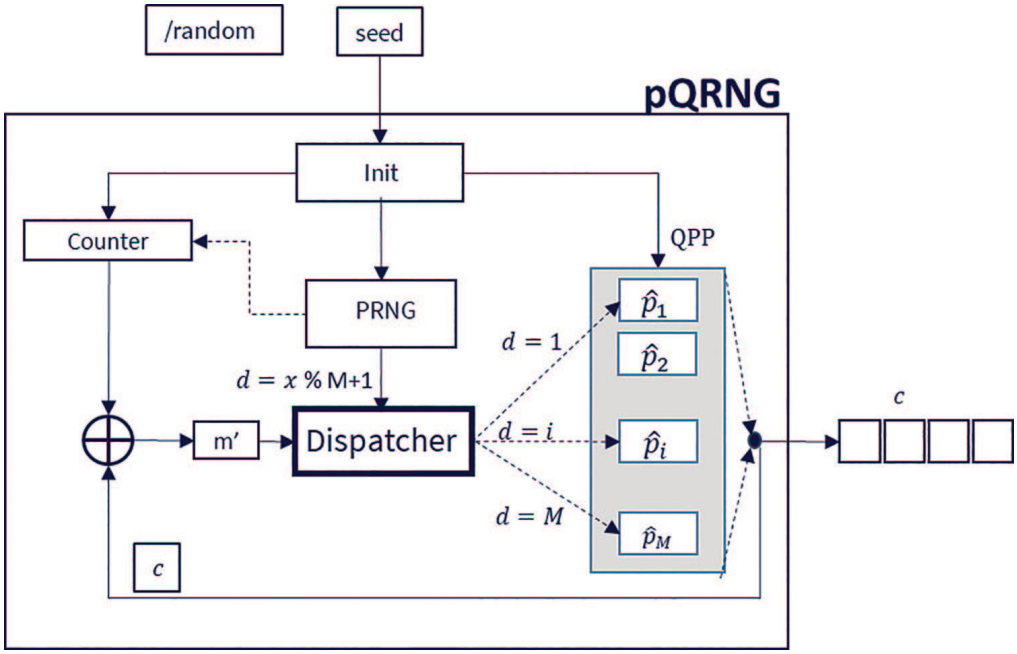

Figure 5 • pQRNG implemented with QPP cryptography [36].

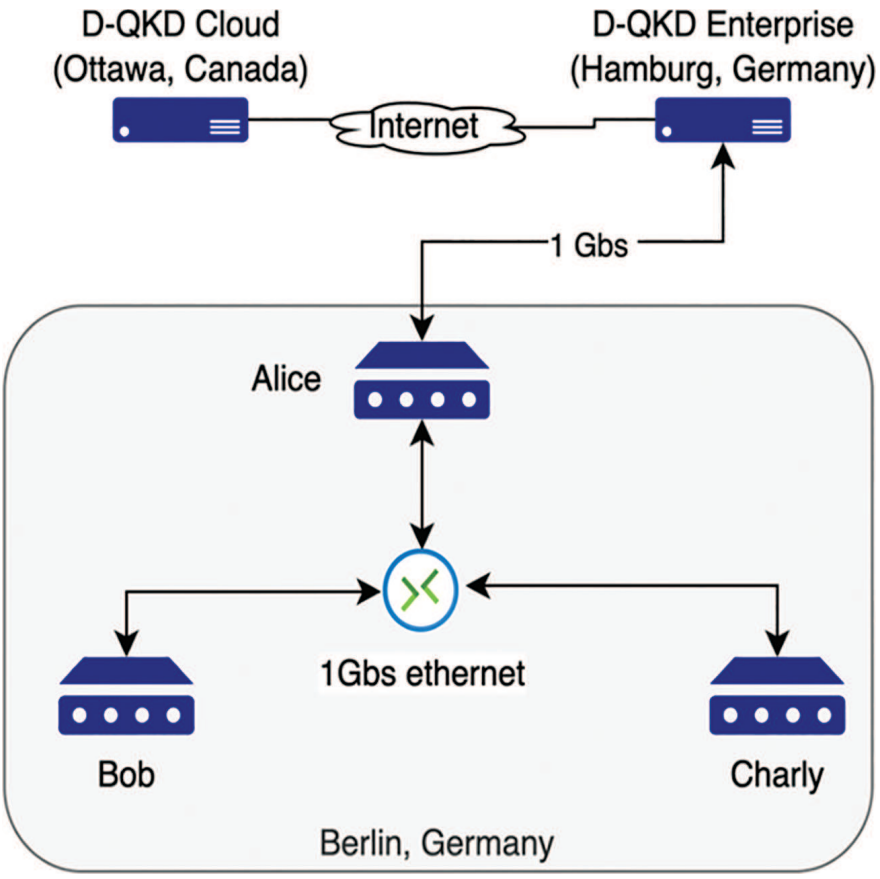

Figure 6 • D-QKD is illustrated as a global quantum random number distribution network [35].

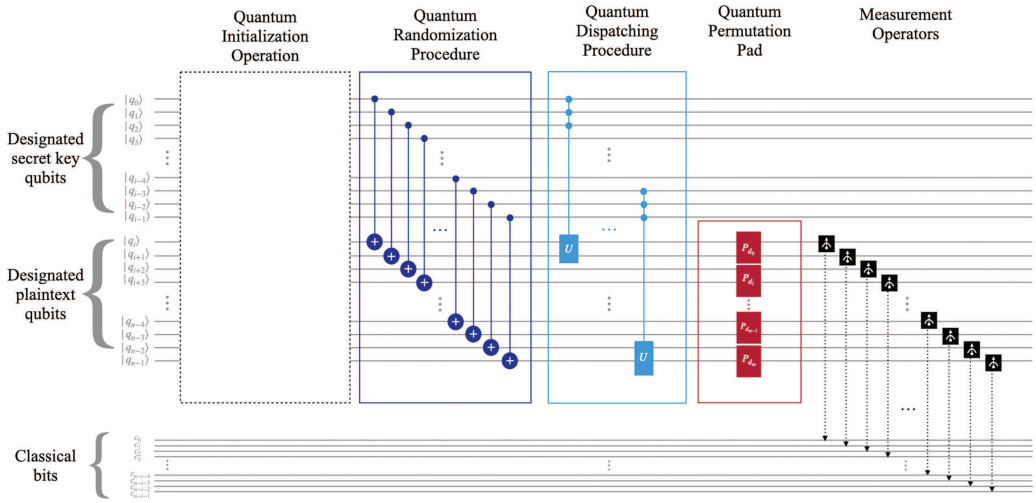

Figure 7 • QPP symmetric cryptography is implemented for encryption in IBMQ [37].

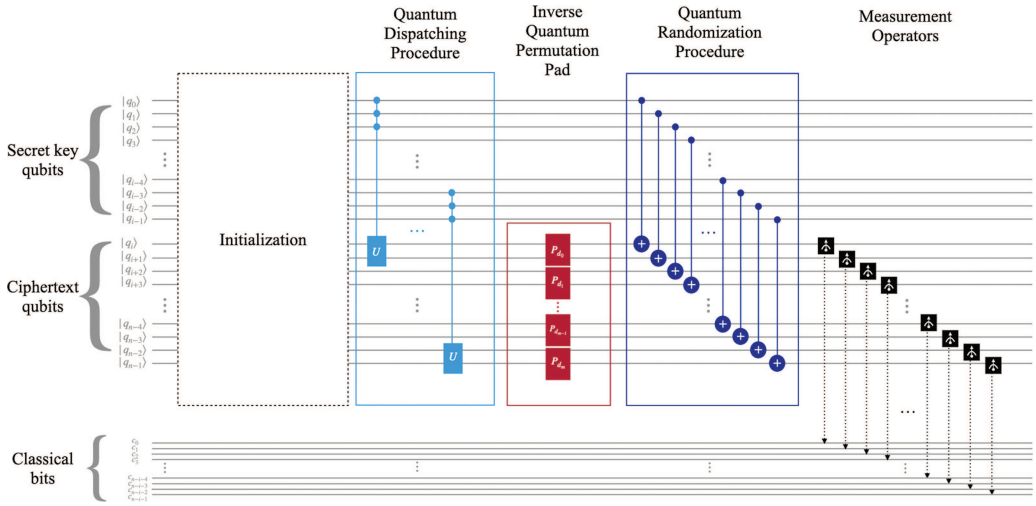

Figure 8 • QPP symmetric cryptography is implemented for decryption in IBMQ [37].

4. Quantum permutation pad asymmetric cryptography

As discussed in Section 2.2.3, permutation operators in QPP-based cryptosystems are represented using modular arithmetic, which exhibits partial homomorphic properties. These properties are particularly advantageous for constructing secure public key cryptographic schemes. More broadly, the algebraic structures of permutation operators and their modular multiplications align well with MPKC, where they contribute to the development of cryptographic protocols such as KEM and DS. This section explores the application of QPP in asymmetric cryptography, emphasizing its role in enhancing security and efficiency within MPKC and related schemes.

One of the earliest examples of public key cryptography using permutation operators is the Merkle–Hellman knapsack cryptosystem (MHC) [41], which used a single permutation operator derived from a coprime pair of integers. While innovative, the Merkle–Hellman cryptosystem was eventually broken due to specific cryptanalytic attacks. To overcome the vulnerabilities of the knapsack cryptosystem, MPKC evolved, shifting from the use of a single multivariate polynomial with a linear mapping over a binary vector space, as in the knapsack system, to the use of multiple multivariate polynomials with bilinear mappings over a finite vector space. This evolution led to cryptographic schemes based on the multivariate quadrature (MQ) problem [70], which provided stronger security. However, the tradeoffs in MPKC schemes include larger public key sizes, increased ciphertext sizes, and lower performance compared to traditional cryptographic systems.

To address these limitations, a new multivariate scheme called homomorphic polynomial public key (HPPK) cryptography was proposed by Kuang and Perepechaenko in 2023 for KEMs [52]. This scheme was later extended for DS [53]. HPPK employs two multivariate polynomials paired with bilinear mappings over a large prime finite field $\mathbb{F}_p$, enhancing the security of both KEM and DS protocols. An essential feature of HPPK is the use of two modular multiplication operators, as shown in Eq. (11), to encrypt the polynomial coefficients. These operators benefit from their partial homomorphic properties, enabling efficient cryptographic operations while maintaining robust security against known attacks.

Compared to MQ-based cryptosystems, HPPK offers several advantages. Its arithmetic QPPs, consisting of modular multiplication operators, provide both confidentiality and integrity in the encrypted data. Additionally, the homomorphic nature of these operators allows cryptographic operations such as encryption, decryption, and signature generation to preserve algebraic structures, making HPPK an attractive choice for PQC.

To provide a thorough understanding of multivariate cryptosystems, we will review three primary types of MPKC schemes. First, we will revisit the MHC (Section 4.1). Next, we will discuss the evolution of multivariate polynomial cryptography and its advancements in MPKC (Section 4.2), with particular emphasis on a proposed QPP-based variant. Finally, we will review HPPK cryptography in detail (Section 4.3), focusing on its use in KEM and DS applications, and its advantages over existing multivariate schemes.

4.1. Merkle–Hellman cryptosystems

The Merkle–Hellman cryptosystem, introduced in 1978 by Ralph Merkle and Martin Hellman [41], is one of the earliest public key cryptographic schemes. It is based on the computational hardness of the subset-sum (or knapsack) problem, a well-known NP-complete problem.

4.1.1. Key generation

The key generation process involves producing a public key for encryption and a private key for decryption. The process is characterized as follows:

Superincreasing sequence

Choose a superincreasing sequence of integers $\{a_1, a_2, \ldots, a_n\}$, where each element is larger than the sum of all preceding elements, that is, $a_i > \sum_{j=1}^{i-1} a_j$. This sequence can be considered as the coefficients of a multivariate polynomial $y(x_1, \ldots, x_n) = \sum_{j=1}^{n} a_j x_j$, with $x_j \in \mathbb{F}_2^n$.

Modulus and multiplier

Select a large modulus M such that $M > \sum_{i=1}^{n} a_i$, and a multiplier W that is coprime to M, that is, $\gcd(W, M) = 1$. A corresponding permutation operator $\hat{p} = W \times \square \mod M$ is applied to the coefficients a_i denoting by the $\square$.

Public key

The public key is computed by applying the permutation operator $\hat{p}$ to each element of the superincreasing sequence:

$$b_i = \hat{p}a_i = Wa_i \bmod M, \quad i = 1, 2, \ldots, n$$

The public key is the sequence $\{b_1, b_2, \ldots, b_n\}$, and the corresponding public polynomial is $y'(x_1, \ldots, x_n) = \sum_{i=1}^{n} b_i x_i$.

Private key

The private key consists of the original superincreasing sequence $\{a_1, a_2, \ldots, a_n\}$, along with the modulus M and the multiplier W.

4.1.2. Encryption

To encrypt a binary message $\mathbf{m} = (x_1, x_2, \ldots, x_n)$, where each $x_i \in \{0, 1\}$, the sender computes the ciphertext C as follows:

$$C = y'(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} b_i x_i$$

The ciphertext C is then transmitted to the recipient.

4.1.3. Decryption

Decryption uses the private key to recover the original message $\mathbf{m}$.

1. **Inverse permutation**: Compute the inverse of the permutation operator:

   $$\hat{p}^{-1} = W^{-1} \times \square \bmod M$$

   Apply $\hat{p}^{-1}$ to the ciphertext C to obtain:

   $$\bar{y} = y(x_1, x_2, \ldots, x_n) = \hat{p}^{-1} C \bmod M$$

2. **Subset-sum decryption**: Using the superincreasing sequence $\{a_1, a_2, \ldots, a_n\}$, recover the binary message $\mathbf{m}$. Since the sequence is superincreasing, there is a unique way to express $\bar{y}$ as a sum of the elements in $\{a_1, a_2, \ldots, a_n\}$. The message bit $x_i = 1$ if a_i is included in the sum, and $x_i = 0$ otherwise.

This decryption process is efficient due to the superincreasing nature of the private key sequence, ensuring that the subset-sum problem can be easily solved during decryption.

4.2. Multivariate public key cryptosystems

Multivariate public key cryptosystems (MPKC) are based on the hardness of solving systems of multivariate quadratic (MQ) equations over finite fields, making them strong candidates for PQC due to their resistance to attacks by quantum computers. In MPKC, the message is defined over a vector space $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q$, where $\mathbb{F}_q$ is a finite field of size q.

4.2.1. Key generation

In a basic MPKC cryptosystem, a set of L bilinear multivariate polynomials, $p_1(\vec{x}), \ldots, p_L(\vec{x})$, govern the encryption and decryption processes. These polynomials are expressed in the following form:

$$p_\ell(x_1, \ldots, x_n) = \vec{x}^T \cdot P_\ell \cdot \vec{x}, \quad \ell = 1, 2, \ldots, L \tag{18}$$

Here, P_ℓ is a private square matrix associated with coefficients of the ℓ-th polynomial, and $\mathbf{P} \in \mathbb{F}_q^{L \times n \times n}$, a compact expression of all L matrices, is called the private *central map* in the MPKC cryptosystem [70]. The central map is quadratic in nature, as each polynomial involves a bilinear form of the input vector $\vec{x}$. The system of equations $\{p_\ell(\vec{x})\}$ forms the core of the cryptographic protocol.

In order to make the central map $\mathbf{P}$ secure, MPKC employs two additional linear maps, typically denoted as $\mathbf{S} \in \mathbb{F}_q^{L \times L}$ and $\mathbf{T} \in \mathbb{F}_q^{n \times n}$. These maps, referred to as the "scrambler" ($\mathbf{S}$) and "affine transformation" ($\mathbf{T}$), obscure the private central map $\mathbf{P}$ and ensure the system's security. The overall encryption process is then expressed as follows:

$$\mathcal{P}(\vec{x}) = \vec{x}^T \cdot [\mathbf{S} \circ \mathbf{P} \circ \mathbf{T}] \cdot \vec{v} = \vec{v}^T \cdot \mathcal{P} \cdot \vec{x} \tag{19}$$

Here, $\mathcal{P} = \mathbf{S} \circ \mathbf{P} \circ \mathbf{T}$ is the public key, while $\mathbf{S}, \mathbf{T}, \mathbf{P}$ constitute the private key. MPKC has a public key size $L \times \frac{n(n+1)}{2} \times \lceil \log_2(q) \rceil$ bits.

4.2.2. Encryption

The encryption process in MPKC involves several steps:

1. **Message representation**: The plaintext message is first encoded into a vector $\vec{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n$.
2. **Applying the public key**: Using the public key $\mathcal{P}$, the polynomials are evaluated at $\vec{x}$, producing a set of values:

$$C_i = \mathcal{P}_i(\vec{x}) = \vec{x}^T \cdot \mathcal{P}_i \cdot \vec{x} \mod q, \quad i = 1, \ldots, L$$

The ciphertext $C = (C_1, \ldots, C_L)$ is then transmitted to the recipient.

The polynomials $\mathcal{P}_i(\vec{x})$ are quadratic, making them computationally hard to invert without knowledge of the private key. This ensures the security of the encrypted message.

4.2.3. Decryption process

The decryption process for the ciphertext $\vec{c} \in \mathbb{F}_q^L$ using the public key $\mathcal{P} = \mathbf{S} \circ \mathbf{P} \circ \mathbf{T}$, where $\mathbf{S}$ and $\mathbf{T}$ are invertible affine maps and $\mathbf{P}$ is the central quadratic map, proceeds as follows:

1. **Apply the inverse of** S: Compute the intermediate value $\vec{z}$ by applying the inverse of the affine map $\mathbf{S}$ to the ciphertext $\vec{c}$:

$$\vec{z} = \mathbf{S}^{-1}(\vec{c})$$

2. **Solve the central quadratic map P**: Solve the system of quadratic equations given by the central map P to find $\vec{y}$:

$$\vec{y} = \mathbf{P}^{-1}(\vec{z})$$

This step involves solving a set of quadratic equations, which is computationally difficult without the private key $\mathbf{P}$.

3. **Apply the inverse of T**: Once $\vec{y}$ is obtained, apply the inverse of the affine map T to recover the original plaintext vector $\vec{x}$:

$$\vec{x} = \mathbf{T}^{-1}(\vec{y})$$

The vector $\vec{x} \in \mathbb{F}_q^n$ represents the decrypted message.

4.2.4. Proposal for multivariate public key cryptosystem with quantum permutation pad

Combining MPKC with QPP encryption is a promising avenue for enhancing cryptographic security while potentially improving operational efficiency. This integration leverages the unique properties of QPP's permutation operators, which can be applied to either the private central map $\mathbf{P}$ or the public central map $\mathcal{P}$ within the MPKC framework.

In traditional MPKC schemes, the security relies on the complexity of solving systems of multivariate polynomial equations. The polynomials are generally represented as follows:

$$p_\ell(x_1, \ldots, x_n) = \vec{x}^T \cdot P_\ell \cdot \vec{x}, \quad \ell = 1, 2, \ldots, L, \tag{20}$$

where $P_\ell \in \mathbb{F}_q^{n \times n}$ is the private reversible central map.

By integrating a secret QPP, we introduce a pad of L arithmetic permutation operators $\hat{p}_\ell = R_\ell \cdot \square \bmod S_\ell$ applied to the private central maps. This allows us to encrypt the central map into a cipher central map:

$$\mathcal{P}_\ell(\vec{x}) = \vec{x}^T \cdot [\hat{p}_\ell P_\ell] \cdot \vec{x} = \vec{v}^T \cdot \mathcal{P}_\ell \cdot \vec{x}, \tag{21}$$

where $\mathcal{P}_\ell = R_\ell P_\ell \bmod S_\ell$ represents the encrypted cipher central map over a hidden ring $[0, S_\ell)$. There is a requirement on moduli S_ℓ to be larger than q^2 for the homomorphic property [52]. With encryption from QPP, two linear affine maps are no longer applicable because they require a modular operation with mod$q$.

The encryption process is the same as traditional MPKC by mapping a secret message over $\mathbb{F}_q$ and obtaining a set of secret segments $\{x_1, x_2, \ldots, x_n\}$. Pre-evaluations on $x_{ij} = x_i x_j \bmod q$ are first performed, and ciphertext can be calculated as follows:

$$C_\ell = \mathcal{P}_\ell(\vec{x}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathcal{P}_{\ell ij} x_{ij} \tag{22}$$

without further mod$q$. The ciphertext is $C = \{C_\ell\}$.

The decryption process has two stages: first decryption with the reverse permutation operators $\hat{p}_\ell^{-1} = R_\ell^{-1} C_\ell \bmod S_\ell \bmod q = c_\ell, \ell = 1, 2, \ldots, L$, and then solving the quadrature equation system associated with the private map $\mathbf{P}$

$$c_\ell = \vec{x}^T \cdot P_\ell \cdot \vec{x}, \quad \ell = 1, 2, \ldots, L, \tag{23}$$

for the plaintext message $\vec{x} = (x_1, x_2, \ldots, x_n)$.

The QPP encryption can be directly applied to traditional MPKC public key for further enhancement of MPKC security, which can further conceal the mathematical structure behind the original MPKC public key. This could be an interesting variant to explore for the MPKC cryptosystem.

4.3. Homomorphic polynomial public key cryptography for key encapsulation mechanism

In contrast to the traditional MPKC [70], Kuang and Perepechaenko introduced a new cryptographic scheme based on a different private central map constructed over two distinct vector spaces. One of these vector spaces is the polynomial vector space $\mathbb{F}_p[x]_{\leq n}$, used for encoding the secret message, and the other is the vector space $\mathbb{F}_p[u_1, \ldots, u_m]$, which introduces noise into the cryptographic process [52]. This combination aims to create a robust KEM suitable for PQC, an area of increasing importance as quantum computing threatens classical cryptographic systems.

The polynomial equations defining the system are given by

$$p_\ell(x, u_1, \ldots, u_m) = \vec{x}^T \cdot P_\ell \cdot \vec{u}, \quad \ell = 1, 2, \ldots, L \tag{24}$$

Here, $\vec{x} = (1, x, x^2, \ldots, x^n)$ represents the vector of powers of the secret message x, while $\vec{u} = (u_1, \ldots, u_m)$ represents a vector of random noise variables. The private central map P_ℓ is constructed as a matrix, specifically designed to introduce complexity in the encryption process. The central map P_ℓ is a matrix with its elements $P_{\ell ij}$, where $\ell = 1, 2$, $i = 0, 1, 2, \ldots, n$, and $j = 1, 2, \ldots, m$. The central map is carefully designed to form an underdetermined multivariate equation system, a feature that enhances the security against certain types of algebraic attacks.

4.3.1. Key generation

One of the key differences between HPPK and traditional MPKC is how the private central map is constructed. In MPKC, the private

central map is typically reversible and randomly selected from a large space. However, in HPPK, Kuang and Perepechaenko [52] proposed a construction based on two univariate polynomials of order λ, $f(x)$ and $h(x)$, combined with a multivariate noise polynomial $B(x, u_1, \ldots, u_m)$ of order n' for variable x. The noise polynomial is defined as follows:

$$B(x, u_1, \ldots, u_m) = \sum_{i=0}^{n'} \sum_{j=1}^{m} b_{ij} x^i u_j \tag{25}$$

This construction introduces additional security by making it difficult for attackers to invert the polynomial system. The central polynomials in the scheme are then defined as follows:

$$\begin{aligned} p_1(x, u_1, \ldots, u_m) &= f(x) B(x, u_1, \ldots, u_m) \\ &= \sum_{j=1}^{m} \sum_{i=0}^{n=\lambda+n'} P_{1ij} x^i u_j \bmod p \\ p_2(x, u_1, \ldots, u_m) &= h(x) B(x, u_1, \ldots, u_m) \\ &= \sum_{j=1}^{m} \sum_{i=0}^{n=\lambda+n'} P_{2ij} x^i u_j \bmod p \end{aligned} \tag{26}$$

with $P_{1ij} = \sum_{s+t=i} f_s b_{tj}$ and $P_{2ij} = \sum_{s+t=i} h_s b_{tj}$. It is obvious from Eq. (26) that

$$\frac{p_1(x, u_1, \ldots, u_m)}{p_2(x, u_1, \ldots, u_m)} = \frac{f(x)}{h(x)} \bmod p \tag{27}$$

This indicates that the multivariate noise polynomials have no impact on the division of two product polynomials in Eq. (26), which offers great potential for random encapsulation with noise variables.

To secure the private central map, a private *QPP* is employed to encrypt the private map. This technique involves the application of two modular multiplication permutation operators:

$$\hat{p}_\ell = R_\ell \cdot \square \bmod S_\ell, \ell = 1, 2 \tag{28}$$

applying to the private central map, P_1 and P_2. Specifically, the encryption is performed as follows:

$$\mathcal{P}_{1ij} = R_1 (P_{1ij} + p) \bmod S_1, \quad \mathcal{P}_{2ij} = R_2 (P_{2ij} + p) \bmod S_2 \tag{29}$$

where extra additions with p are used to avoid too small integers $P_{\ell ij} \in \mathbb{F}_p$, together with a condition of $R_1 > \frac{S_1}{p}$ and $R_2 > \frac{S_2}{p}$. Equation (29) forms the HPPK public key for KEM. The key pair consists of the following:

- **Public key**:
  - $\mathcal{P}_{1ij}, \mathcal{P}_{2ij}$ for $i = 0, 1, \ldots, n = \lambda + n', j = 1, 2, \ldots, m$.
- **Private key**:
  - Univariate polynomials: $f(x), h(x)$
  - QPP: $\hat{p}_1 \leftarrow (R_1, S_1); \hat{p}_2 \leftarrow (R_2, S_2)$.

4.3.2. Encapsulation

The encryption (encapsulation) process begins with the public keys $\mathcal{P}_1$ and $\mathcal{P}_2$ from Eq. (29). To encrypt a secret message x chosen randomly from $\mathbb{F}_p$, the encryptor follows these steps:

- **Noise selection and preparation**: Choose random noise values $u_1, \ldots, u_m \in \mathbb{F}_p$. Compute $x_{ij} = x^i u_j \bmod p$ for $i = 0, 1, \ldots, n$ and $j = 1, 2, \ldots, m$.
- **Polynomial evaluation**:

$$C_1 = \sum_{i=0}^{n} \sum_{j=1}^{m} \mathcal{P}_{1ij} x_{ij}, \quad C_2 = \sum_{i=0}^{n} \sum_{j=1}^{m} \mathcal{P}_{2ij} x_{ij}$$

- **Ciphertext formation**: The ciphertext is formed as $CT = \{C_1, C_2\}$.

In this process, no modular operations are required for the encryptor, as the moduli S_1 and S_2 are parts of the private key and thus unknown to the encryptor.

4.3.3. Decapsulation

Upon receiving the ciphertext $CT = \{C_1, C_2\}$, the decrypter (who knows the private key) proceeds as follows:

- **Symmetric decryption**:

$$c_1 = \left(\frac{C_1}{R_1} \bmod S_1 \right) \bmod p, \quad c_2 = \left(\frac{C_2}{R_2} \bmod S_2 \right) \bmod p$$

- **Noise elimination**:

$$k = \frac{c_1}{c_2} \bmod p = \frac{f(x)}{h(x)} \bmod p$$

- **Secret extraction**: Solve $f(x) - kh(x) = 0$ for x.

If $\lambda = 1$ (a linear system), the secret x can be efficiently recovered using

$$x = \frac{kh_0 - f_0}{f_1 - kh_1} \bmod p$$

By using QPP encryption and modular operations, HPPK offers a strong level of security that complicates potential algebraic attacks. The combination of multivariate and univariate systems, along with noise terms, ensures its resilience.

4.4. Homomorphic polynomial public key cryptography for digital signature

In the HPPK KEM, decryption relies on the reverse permutation operators, $\hat{p}_1$ and $\hat{p}_2$, defined as $\hat{p}_1 = R_1 \cdot \square \bmod S_1$ and $\hat{p}_2 = R_2 \cdot \square \bmod S_2$, where $\square$ represents certain integers used in the encryption process. These operators incorporate the private moduli S_1 and S_2, and their security stems from the fact that they are non-commutative with other operators that use different moduli. Any attempt to manipulate the system leads to re-encryption due to this non-commutativity, forcing attackers into an endless loop of redundant encryptions. Consequently, brute-force search remains the feasible attack strategy.

However, the situation changes when the KEM scheme is extended to a DS scheme. In the case of DS, the verifier needs access to the moduli S_1 and S_2 to perform signature verification. These moduli are private, meaning that their disclosure would compromise the security of the system. To resolve this issue, Kuang et al. recently proposed the use of the Barrett transformation, which

eliminates the need for the verifier to have access to the private moduli during the verification process [53]. In this section, we will systematically review the HPPK DS scheme, focusing on its correctness and security derived from the combination of QPP symmetric encryption and the Barrett transformation.

4.4.1. Correctness

The correctness of the HPPK DS scheme is based on the decryption equation from the HPPK KEM, as shown in Eq. (27). The verification equation evolves as follows:

$$\begin{aligned} \left[\bar{f}p_2(x,\vec{u})\right] \bmod p &= \left[\bar{h}p_1(x,\vec{u})\right] \bmod p \\ \left[\bar{f}R_2^{-1}R_2p_2(x,\vec{u}) \bmod S_2\right] \bmod p &= \left[\bar{h}R_1^{-1}R_1p_1(x,\vec{u}) \right. \\ &\quad \left. \bmod S_1\right] \bmod p \\ \rightarrow \sum_{i=0}^{n}\sum_{j=1}^{m}(F\mathcal{P}_{2ij} \bmod S_2)x^iu_j &= \sum_{i=0}^{n}\sum_{j=1}^{m}(H\mathcal{P}_{1ij} \bmod S_1)x^iu_j \bmod p \\ \rightarrow \sum_{i=0}^{n}\sum_{j=1}^{m}V_{ij}x^iu_j &= \sum_{i=0}^{n}\sum_{j=1}^{m}U_{ij}x^iu_j \bmod p \end{aligned} \tag{30}$$

where $\bar{f} = f(x)$ and $\bar{h} = h(x)$ are the plaintext signatures. The signature elements, $F = \bar{f}R_2^{-1} \bmod S_2$ and $H = \bar{h}R_1^{-1} \bmod S_1$, form the signature $Sig = \{F, H\}$, encrypted with the permutation operators. The coefficients V_{ij} and U_{ij} are defined as follows:

$$U_{ij} = H\mathcal{P}_{1ij} \bmod S_1, \quad V_{ij} = F\mathcal{P}_{2ij} \bmod S_2 \tag{31}$$

However, since the private moduli S_1 and S_2 remain unknown to the verifier, they cannot directly perform verification based on Eq. (30). To address this, Kuang et al. proposed using the Barrett reduction algorithm with a Barrett parameter $R = 2^K$, where the ideal case sets $K = 2L$ (practically, $K = L + 32$ is often sufficient). With the Barrett transformation, the coefficients U_{ij} and V_{ij} can be computed as follows:

$$\begin{aligned} V_{ij} &= \beta(F\mathcal{P}_{2ij} \bmod S_2) \bmod p = Fp'_{2ij} - s_2\left\lfloor\frac{F\nu_{ij}}{R}\right\rfloor \bmod p, \\ U_{ij} &= \beta(H\mathcal{P}_{1ij} \bmod S_1) \bmod p = Hp'_{1ij} - s_1\left\lfloor\frac{H\mu_{ij}}{R}\right\rfloor \bmod p \end{aligned} \tag{32}$$

where β is a randomly chosen element from the field $\mathbb{F}_p$. The public key for verification, PK_v, is now shifted from the public key for encapsulation, PK_e in HPPK KEM:

$$\mathcal{P}_{1ij} = R_1P_{1ij} \bmod S_1, \quad \mathcal{P}_{2ij} = R_2P_{1ij} \bmod S_2 \tag{33}$$

to HPPK DS:

$$\begin{aligned} p'_{2ij} &= \beta\mathcal{P}_{2ij} \bmod p, \quad p'_{1ij} = \beta\mathcal{P}_{1ij} \bmod p, \\ \nu_{ij} &= \left\lfloor\frac{R\mathcal{P}_{2ij}}{S_2}\right\rfloor, \quad \mu_{ij} = \left\lfloor\frac{R\mathcal{P}_{1ij}}{S_1}\right\rfloor, \\ s_1 &= \beta S_1 \bmod p, \quad s_2 = \beta S_2 \bmod p \end{aligned} \tag{34}$$

This eliminates the need for private moduli S_1 and S_2 in the verification equation. The verification equation now follows Eq. (30), with coefficients defined by Eq. (32). It is important to note that for security purposes, the random multivariate polynomial $B(x,\vec{u})$ used during public key generation in HPPK DS must differ from the one used in the KEM scheme.

As shown earlier, both polynomials $U(x,\vec{u})$ and $V(x,\vec{u})$ are nonlinearly determined by the received signature $Sig = \{F, H\}$, significantly complicating potential forgery attacks.

4.4.2. Signing

To sign a message M using HPPK DS, the signer first hashes the message with a cryptographic hash function such as SHA256, SHA384, or SHA512, resulting in $m = \text{HASH}(M)$. If the bit-length of the hash $|m|_2$ exceeds the bit-length of the prime $|p|_2$, the message is divided into segments over the field $m = \{m_k \in \mathbb{F}_p\}$. For each segment $x = m_k$, the signer follows these steps to generate a signature:

- Selects a random $\alpha \in \mathbb{F}_p$,
- Computes the polynomials $\bar{f} = \alpha\sum_{i=0}^{\lambda}f_ix^i \bmod p$ and $\bar{h} = \alpha\sum_{i=0}^{\lambda}h_ix^i \bmod p$,
- Computes the signature elements $F = R_2^{-1}\bar{f} \bmod S_2$ and $H = R_1^{-1}\bar{h} \bmod S_1$.

The signer then concatenates all segments of the signature to form the complete signature $\text{Sig} = \{F, H\}$, which is sent along with the message to the verifier.

4.4.3. Verification

To verify the signature, the verifier applies the same cryptographic hash function and segmentation process as the signer. For each segment signature F_k, H_k corresponding to the hash segment $x = m_k$, the verifier calculates the following coefficients:

$$\begin{aligned} V_{ij} &= F_kp'_{2ij} - s_2\left\lfloor\frac{F_k\nu_{ij}}{R}\right\rfloor \bmod p, \\ U_{ij} &= H_kp'_{1ij} - s_1\left\lfloor\frac{H_k\mu_{ij}}{R}\right\rfloor \bmod p \end{aligned} \tag{35}$$

For each noise variable u_j, the verifier then evaluates

$$V_j(x) = \sum_{i=0}^{n}V_{ij}x^i \bmod p, \quad U_j(x) = \sum_{i=0}^{n}U_{ij}x^i \bmod p$$

The verification passes if $V_j(x) = U_j(x)$ for all segments; otherwise, it fails.

4.5. Advances in cryptanalysis

4.5.1. Quantum permutation pad cryptography

In the context of QPP symmetric cryptography, Kuang and Barbeau thoroughly analyzed its security, demonstrating resilience against various attacks, including chosen plaintext and chosen ciphertext attacks [32]. Key to maintaining this security is proper implementation of the QPP scheme, which includes using a randomly chosen QPP, pre-randomizing the plaintext, and employing random dispatching to different permutation gates for each n-bit segment. These measures enhance confusion and diffusion, which are essential properties for preventing pattern recognition in the ciphertext.

In 2023, Amil and Gupta attempted a limited cryptanalysis of QPP, focusing on bit position permutations within plaintext bit strings, which did not fully address the complexity of the QPP scheme [55].

In 2024, Zawadzki presented a more sophisticated chosen-plaintext attack by simplifying the QPP process, exposing potential vulnerabilities [71]. By removing the pre-randomization step,

he demonstrated the theoretical possibility of identifying the underlying permutation matrices used in the encryption. However, the random dispatching of permutation matrices still introduces significant complexity.

Zawadzki also combined his attack with side-channel analysis, aiming to intercept the intermediate ciphertext before random dispatching to reveal the PRNG value governing the encryption. However, the cryptographic strength of PRNGs, derived from the hash of a shared secret key, complicated this attack. Zawadzki concluded that his chosen-plaintext attack alone was insufficient against a properly implemented QPP scheme.

To address the vulnerabilities, an enhancement to the QPP process involves incorporating a random initial vector (IV). XORing the hash of the shared key with a randomly generated IV refreshes the PRNG seed for each encryption, ensuring varied dispatching sequences even with identical plaintexts. This additional randomness enhances security, making Zawadzki's attack unviable.

Including the IV in the ciphertext is crucial for accurate decryption. To reduce overhead, a session ID can serve as a lightweight substitute for the IV, minimizing additional data transmission. This enhancement effectively mitigates chosen-plaintext attacks, reinforcing QPP symmetric cryptography against advanced cryptanalytic techniques.

4.5.2. Merkle–Hellman cryptosystem

The Merkle–Hellman cryptosystem [41] exhibits two significant vulnerabilities: its dependence on a superincreasing sequence and its linear binary encoding through a linear mapping. The first vulnerability, the superincreasing sequence, can be partially mitigated using techniques such as secret permutations [72], the Chinese remainder theorem [73], and other methods [74]. Nevertheless, these techniques do not fully eliminate the inherent weaknesses of the superincreasing structure, as highlighted in various studies [75, 76].

The second vulnerability, associated with the linear binary encoding, leads to the formulation of a classical knapsack problem, which is particularly susceptible to lattice reduction attacks. As demonstrated in the literature, any variations of the Merkle–Hellman cryptosystem that retain this encoding mechanism remain vulnerable to such attacks [75, 76]. Consequently, while the cryptosystem has seen adaptations, the core issues stemming from the encoding approach continue to pose security risks. This highlights the necessity for exploring alternative encoding schemes or structural changes to enhance resilience against cryptanalytic attacks.

4.5.3. Multivariate public key cryptography cryptosystem

Multivariate public-key cryptosystems (MPKCs) are promising candidates for PQC, based on the NP-hard problem of solving MQ equations over finite fields. Despite their resilience, MPKC is vulnerable to several cryptanalytic methods.

Gröbner basis attacks

The main technique for cryptanalyzing MPKC involves Gröbner basis methods, which solve quadratic systems more efficiently than brute force.

- *Improvements in F4/F5 algorithms:* Enhancements have increased the efficiency of attacks by exploiting structural weaknesses in schemes like HFE and UOV [77, 78].

- *Variants of Gröbner basis attacks:* Novel approaches, including *sparse elimination* and *XL (extended linearization)* algorithms, exploit the algebraic structure of polynomials in specific MPKC schemes [79].

Rank attacks and the MinRank problem

Rank-based attacks, especially the MinRank problem, significantly impact MPKC schemes such as UOV and Rainbow [80, 81].

- *MinRank problem:* Recent advancements in low-rank recovery algorithms have compromised some MPKC schemes, notably those using large field extensions or structures like Rainbow [80].

- *High-rank UOV schemes:* Research is ongoing to develop high-rank UOV variants [82], although these have not been thoroughly tested against emerging MinRank techniques.

Differential and rank-based cryptanalysis

Differential cryptanalysis has been adapted to multivariate cryptosystems, employing techniques like differential-algebraic attacks to exploit statistical relationships in quadratic maps.

- *Differential-algebraic attacks:* These identify differential invariants to reduce the number of required equations for successful attacks against structured systems like HFE and Rainbow [83–85].

- *Rank-based differential attacks:* Combining rank and differential methods has led to enhanced techniques for attacking systems with specific structures, notably Rainbow-type schemes [86–88].

Utilizing arithmetic QPP or modular multiplication over large hidden rings, as discussed in Section 4.2.4, may significantly enhance MPKC security. This approach obscures relationships between elements, complicating traditional algebraic and differential attacks. By transforming the MQ problem into an underdetermined system, it preserves the structural integrity of the cryptographic scheme, enhancing resistance against common cryptanalytic methods that exploit symmetries in finite fields.

4.5.4. Homomorphic polynomial public key cryptography

HPPK KEM

The HPPK KEM [52] introduces a distinctive framework for conducting private key recovery attacks. Central to this approach is a system of polynomial equations characterized by large integer coefficients derived from the private moduli S_1 and S_2. These hidden moduli play a crucial role in formulating the equations that define the relationships among the secret parameters.

To analyze private key recovery attacks based on the known public key, the HPPK KEM scheme can be expressed by the system of equations:

$$P_i = R \cdot p_i \mod S, \quad i = 1, 2, \ldots, n \tag{36}$$

Table 2 • The key and cipher sizes in bytes, as provided by the HPPK KEM scheme for the proposed parameter sets, are determined based on the optimal complexity of $\mathcal{O}(2^L)$ with $L = 2\lceil log_2 p \rceil + 16$. All data are presented in bytes. The configuration is defined as $(\log_2 p, L)$ with $n = 1, \lambda = 1, m = 2$

Security	p	Configuration	Entropy (bits)	*PK*	*SK*	*CT*
I	$2^{64} - 59$	(64, 144)	144	**216**	104	216
III	$2^{96} - 17$	(96, 208)	208	**312**	152	234
V	$2^{128} - 159$	(128, 272)	272	**408**	200	204

where $p_i \in \mathbb{F}_p$, $S \gg p^2$, and $R > \frac{S}{p}$. If the sequence p_i forms a **superincreasing sequence**, the problem resembles the **knapsack problem**, which was famously broken by Shamir in 1984 [47]. Shamir's attack involved identifying a forged coprime pair (W', M') that allowed the public key to be decrypted back into a forged superincreasing set, facilitating decryption within the original **Merkle–Hellman scheme** [41]. The binary message encoding utilized in Merkle–Hellman exploited this structural vulnerability.

In contrast, the HPPK KEM employs a **random set of integers** $p_i \in \mathbb{F}_p$ instead of a superincreasing sequence, effectively thwarting the efficacy of the Shamir attack. The reliance on creating a forged coprime pair to reproduce a superincreasing sequence becomes infeasible when p_i is randomized. Consequently, the complexity of brute-force recovery increases significantly. Specifically, searching for a coprime pair to generate a sequence p_i within the range $[p, 2p)$ involves a complexity of $\mathcal{O}(2^{2L})$, as detailed in [52].

The complexity of key recovery attacks using the HPPK KEM public key could be further optimized with the following strategy:

$$\eta \frac{P_i}{P_j} \bmod S' - p \in \mathbb{F}_p, \quad i \neq j, \quad i,j = 1, 2, \ldots, n \tag{37}$$

This approach involves guessing $\eta \in \mathbb{F}_p$ and $S' \in [2^{L-1}, 2^L)$. Once Eq. (37) is satisfied, the candidate p_j can be identified as $p_j = \eta$, and the candidate for S can be determined as $S = S'$, from which R can be derived using $R = \frac{P_j}{p_j} \bmod S$. This improved attack strategy yields a complexity of $\mathcal{O}(p \cdot 2^L)$, a significant enhancement over the simpler brute-force complexity required for finding a potential coprime pair (R, S).

To further optimize the complexity of the key recovery attack using the HPPK KEM public key, one can rewrite Eq. (36) as follows:

$$R \cdot p_i = P_i + k_i \cdot S \quad \rightarrow \quad R \cdot p_i - P_i = k_i \cdot S \tag{38}$$

where k_i are unknown integers for $i = 1, 2, \ldots, n$. This formulation presents a linear equation system with the known public key P_i and the range of secrets: $p_i \in [p, 2p)$.

Applying lattice reduction techniques to the left-hand side allows for the exploration of the space defined by the vector $\mathbf{v} = (p_1, p_2, \ldots, p_n)$. By treating the problem as a search for short vectors in a lattice formed by the equations by guessing R, we can systematically reduce the complexity of solving for k_i for $i = 1, 2, \ldots, n$. After identifying these values, the entire private key can be reconstructed. This process leverages the geometric properties of lattices, which often reveal relationships between numbers that are not immediately apparent in the integer domain.

If this process is successful, the optimal complexity is $\mathcal{O}(2^L)$. However, it is important to note that lattice reduction can be quite challenging in practice. Therefore, we can assume a worst-case scenario from a security perspective, leading to a complexity of $\mathcal{O}(2^L)$ for the key recovery attack on the HPPK KEM.

Regarding secret recovery attacks, the system of polynomial equations can be expressed as follows:

$$\sum_{j=1}^{m} \sum_{i=0}^{n} P_{ij}(x^i u_j \bmod p) = C_1; \quad \sum_{j=1}^{m} \sum_{i=0}^{n} Q_{ij}(x^i u_j \bmod p) = C_2, \tag{39}$$

where $P_{ij} \in S_1$ and $Q_{ij} \in S_2$ are associated with the private key sets S_1 and S_2, and $x, u_1, u_2, \ldots, u_m$ are elements from the finite field $\mathbb{F}_p$. This equation describes a typical **Diophantine problem**, characterized by a search complexity of $O(p^{m+1})$ for brute-force exploration of the solution space. The presence of modular reductions and non-linear terms introduces additional computational complexity, potentially impacting the efficiency of the search.

Equation (39) can be reformulated into a modular knapsack problem:

$$\sum_{j=1}^{m} \sum_{i=0}^{n} P_{ij} x^i u_j = C_1 \bmod p; \quad \sum_{j=1}^{m} \sum_{i=0}^{n} Q_{ij} x^i u_j = C_2 \bmod p.$$

By applying the Gaussian elimination, we can reduce the system to

$$\sum_{j=1}^{m-1} \sum_{i=0}^{2n} d_{ij} x^i u_j = c \bmod p.$$

This reduction results in a complexity of $O(p^{m-1})$ for finding non-deterministic solutions, yielding a solution space complexity of $O(p^{m-1})$.

In summary, the overall deterministic attack is $\mathcal{O}(2^L)$ from the key recovery attack and $\mathcal{O}(p^{m+1})$ from the secret recovery attack. The security requires $m \geq 2$, leading to the latest optimal complexity of $\mathcal{O}(2^L)$ for the HPPK KEM. Based on this updated complexity, we tabulated the updated key and cipher sizes in **Table 2**. Although these sizes are bigger than before, they remain remarkably compact..

Homomorphic polynomial public key digital signature

The Barrett reduction in HPPK DS [53] functions as a non-linear encryption mechanism for symmetrically encrypted polynomial coefficients. This is expressed as follows:

$$\nu_{ij} = \left\lfloor \frac{RQ_{ij}}{S_2} \right\rfloor, \quad \mu_{ij} = \left\lfloor \frac{RP_{ij}}{S_1} \right\rfloor. \tag{40}$$

Here, P_{ij} and Q_{ij} are encrypted using S_1 and S_2, respectively, and their ciphertexts are their quotients. This non-linear property complicates the attacker's ability to derive the encryption key.

For key recovery attacks, Kuang et al. [53] proposed an iterative search algorithm based on the following equations:

$$\begin{aligned} \mu_{ij} &= \left\lfloor \frac{R \cdot P_{ij}}{S_1} \right\rfloor \implies P_{ij} = \left\lceil \frac{S_1 \cdot \mu_{ij}}{R} \right\rceil \\ \nu_{ij} &= \left\lfloor \frac{R \cdot Q_{ij}}{S_2} \right\rfloor \implies Q_{ij} = \left\lceil \frac{S_2 \cdot \nu_{ij}}{R} \right\rceil . \end{aligned} \tag{41}$$

This refined attack iteratively searches for S_1 within the range from 2^{L-1} to 2^L, computing P_{ij} using the known public key μ_{ij}, and recalculating $\mu'_{ij} = \left\lfloor \frac{R \cdot P_{ij}}{S_1} \right\rfloor$. If $\mu'_{ij} = \mu_{ij}$, the correct values of S_1 and all P_{ij} are identified. A similar approach is applicable to the public key ν_{ij}. The overall complexity of this attack is $\mathcal{O}(2^L)$.

Given the public keys p'_{ij}, q'_{ij}, s_1, and s_2, an attacker can find the random $\beta \in \mathbb{F}_p$ without significant computational effort, enabling the retrieval of the entire p_{ij} and q_{ij}. With knowledge of P_{ij}, p_{ij}, and S_1, R_1 can be directly calculated. A similar process applies to obtain R_2. Modular polynomial factorization can finalize $f_\lambda(x)$ and $h_\lambda(x)$ with a common scale factor.

The complexity of creating a forged signature is $\mathcal{O}(2^{2L})$. The Barrett parameter $R = 2^K$ affects public key size, optimizing verification success while minimizing size. For instance, at NIST security level I, public key sizes range from 196 to 356 bytes based on selected K values [53].

Homomorphic polynomial public key triple

The HPPK cryptosystem, used for both KEM and DS, operates with a security complexity of $\mathcal{O}(2^L)$, considering the latest update on the key recovery attack of HPPK KEM in this review. This framework allows for the generation of a structured key triple, consisting of a private key SK, an encapsulation public key PK_e, and a verification public key PK_v.

The private key SK is defined as $f[\lambda+1], h[\lambda+1], (R_1, S_1), (R_2, S_2)$, which are essential parameters for both encryption and signing operations. To generate the encapsulation public key PK_e, a randomized multivariate polynomial $B_e(x, u_1, \ldots, u_m)$ is applied to the private key:

$$PK_e \leftarrow SK + B_e(x, u_1, \ldots, u_m),$$

where $B_e(x, u_1, \ldots, u_m)$ introduces randomness into the encapsulation process. Similarly, for verification purposes, the public key PK_v is derived from the private key using another randomized polynomial $B_v(x, u_1, \ldots, u_m)$:

$$PK_v \leftarrow SK + B_v(x, u_1, \ldots, u_m).$$

With knowledge of SK, the key generator can decrypt any ciphertext encrypted with PK_e, as well as sign messages, which can be verified by anyone holding the corresponding PK_v.

This construction allows the same SK to generate multiple unique pairs of (PK_e, PK_v) by employing different instances of the random polynomials $B_e(.)$ and $B_v(.)$. Each unique pair can be designated for a specific group or session, adding flexibility and security to the cryptosystem.

5. Conclusions

This review has explored the pivotal role of the QPP in advancing the field of quantum-secure cryptography. By leveraging the non-commutative nature of QPP, its applicability across a range of cryptographic systems—such as matrix-based symmetric encryption, quantum gate-based encryption, and arithmetic-based multivariate public key schemes—has been thoroughly examined. Specifically, we have demonstrated how QPP enhances the security of HPPK's KEM and DS schemes, as well as the Merkle–Hellman and MPKC systems, underscoring its potential to resist quantum attacks.

The integration of QPP into these systems not only improves quantum resistance but also brings substantial benefits in terms of key size optimization, cryptographic complexity, and overall security robustness. These advancements position QPP as a critical tool in the development of quantum-safe cryptographic protocols.

Moving forward, practical implementation and benchmarking of QPP's performance across diverse cryptographic applications will be crucial. Testing its efficiency, scalability, and real-world viability can offer further insights. In addition, as quantum computing technologies continue to evolve, investigating how QPP can synergize with other quantum-resistant approaches may lead to even more significant breakthroughs, solidifying its place as a foundational component of quantum-secure cryptography.

Acknowledgments

The author acknowledges Daniel Johnson from Carleton University for his valuable insights during discussions on lattice reduction.

Funding

The author declares no direct financial support was received for the research, authorship, or publication of this article. This work was conducted as part of the author's role at Quantropi Inc.

Author contributions

The author confirms sole responsibility for this work. The author approves of this work and takes responsibility for its integrity.

Conflict of interest

The author declare no conflict of interest.

Competing interest

The author declare that they have no competing interests.

Data availability statement

Data supporting these findings are available within the article, at https://doi.org/10.20935/AcadQuant7457, or upon request.

Institutional review board statement

Not applicable.

Informed consent statement

Not applicable.

Additional information

Received: 2024-10-16

Accepted: 2024-12-10

Published: 2025-01-03

Academia Quantum papers should be cited as *Academia Quantum 2025*, ISSN 3064-979X, https://doi.org/10.20935/AcadQuant7457. The journal's official abbreviation is *Acad. Quant.*

Publisher's note

Academia.edu Journals stays neutral with regard to jurisdictional claims in published maps and institutional affiliations. All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Copyright

©2024 copyright by the authors. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

References

1. Avanzi R, Bos J, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, et al. CRYSTALS-KYBER. Specification document (update from August 2021); 2020 [cited 2024 Nov 27]. Available from: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf

2. Aragon N, Barreto P, Bettaieb S, Bidoux L, Blazy O, Deneuville JC, et al. Bit flipping key encapsulation; 2022 [cited 2024 Nov 27]. Available from: https://bikesuite.org/files/v5.0/BIKE_Spec.2022.10.04.1.pdf

3. Melchor CA, Aragon N, Bettaieb S, Bidoux L, Blazy O, Deneuville JC, et al. Hamming quasi-cyclic (hqc); 2021 [cited 2024 Nov 27]. Available from: http://pqc-hqc.org/doc/hqc-specification_2021-06-06.pdf

4. McEliece RJ. A public-key cryptosystem based on algebraic coding theory. Deep Space Netw Prog Rep. 1978;44:114–6. [cited 2024 Sept 24]. Available from: https://tda.jpl.nasa.gov/progress_report/42-44/44N.PDF

5. Fouque PA, Hoffstein J, Kirchner P, Lyubashevsky V, Pornin T, Prest T, et al. Falcon: fast-fourier lattice-based compact signatures over NTRU (specification v1.2); 2020 [cited 2024 Nov 27]. Available from: https://falcon-sign.info/falcon.pdf

6. Lyubashevsky V, Ducas L, Kiltz E, Lepoint T, Schwabe P, Seiler G, et al. CRYSTALS-dilithium - algorithm specifications and supporting documentation (version 3.1); 2020 [cited 2024 Nov 27]. Available from: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf

7. Aumasson JP, Bernstein DJ, Beullens W, Dobraunig C, Eichlseder M, Fluhrer S, et al. SPHINCS+: specification document (part of the submission package); 2020 [cited 2024 Nov 27]. Available from: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf

8. NIST. Status report on the third round of the nist post-quantum cryptography standardization process; 2022 [cited 2024 Nov 27]. Available from: https://csrc.nist.gov/publications/detail/nistir/8413/final

9. Bernstein DJ, Chuengsatiansup C, Lange T, van Vredendaal C. NTRU prime: reducing attack surface at low cost. In: Adams C, Camenisch J, editors. Selected areas in cryptography – SAC 2017. Cham: Springer International Publishing; 2018. p. 235–60. ISBN 978-3-319-72565-9.

10. D'Anvers JP, Karmakar A, Roy SS, Vercauteren F. MLWR-based kem. 2024 [cited 2023 Nov 1]. Available from: https://www.esat.kuleuven.be/cosic/pqcrypto/saber/index.html

11. NIST. Nist releases first 3 finalized post-quantum encryption standards; 2024 [cited 2024 Aug 13]. Available from: https://www.federalregister.gov/documents/2024/08/14/2024-17956/announcing-issuance-of-federal-information-processing-standards-fips-fips-203-module-lattice-based

12. Bennett CH, Brassard G. Quantum cryptography: public key distribution and coin tossing. Theor Comput Sci. 2014;560:7–11. doi: 10.1016/j.tcs.2014.05.025

13. Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Commun ACM. 1978;21(2):120–6. doi: 10.1145/359340.359342

14. Diffie W, Hellman M. New directions in cryptography. IEEE Trans Inf Theory. 1976;22(6):644–54. doi: 10.1109/TIT.1976.1055638

15. Menezes AJ, Okamoto T, Vanstone SA. Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Trans Inf Theory. 1993;39(5):1639–46. doi: 10.1109/18.259647

16. Shor PW, Preskill J. Simple proof of security of the bb84 quantum key distribution protocol. Phys Rev Lett. 2000;85(2):441–4. doi: 10.1103/physrevlett.85.441

17. Renner R, Gisin N, Kraus B. Information-theoretic security proof for quantum-key-distribution protocols. Phys Rev A. 2005;72:012332. doi: 10.1103/PhysRevA.72.012332

18. Djordjevic IB. Discrete variable (DV) QKD. In: Physical-layer security and quantum key distribution. Cham: Springer Nature Switzerland AG; 2019. ISBN 978-3-030-27564-8.

19. Lai JS, Lin XY, Qian Y, Liu L, Zhao WY, Zhang HY. Deployment-oriented integration of dv-qkd and 100 g optical transmission system. In Asia Communications and Photonics Conference (AC PC) 2019, Chengdu. Optical Society of America; 2019. p. T2H.1. [cited 202 Nov 27]. Available from: http://opg.optica.org/abstract.cfm?URI=ACPC-2019-T2H.1

20. Pirandola S, Mancini S, Lloyd S, Braunstein SL. Continuous-variable quantum cryptography using two-way quantum communication. Nat Phys. 2008;4(9):726–30. doi: 10.1038/nphys1018

21. Pirandola S, García-Patrón R, Braunstein SL, Lloyd S. Direct and reverse secret-key capacities of a quantum channel. Phys Rev Lett. 2009;102(5). doi: 10.1103/physrevlett.102.050503

22. Weedbrook C, Pirandola S, García-Patrón R, Cerf NJ, Ralph TC, Shapiro JH, et al. Gaussian quantum information. Rev Mod Phys. 2012;84(2):621–69. doi: 10.1103/revmodphys.84.621

23. Lucamarini M, Yuan ZL, Dynes JF, Shields AJ. Overcoming the rate-distance limit of quantum key distribution without quantum repeaters. Nature. 2018;557(7705):400–3. doi: 10.1038/s41586-018-0066-6

24. Lu FY, Yin ZQ, Wang R, Fan-Yuan GJ, Wang S, He DY, et al. Practical issues of twin-field quantum key distribution. New J Phys. 2019;21(12):123030. doi: 10.1088/1367-2630/ab5a97

25. Minder M, Pittaluga M, Roberts GL, Lucamarini M, Dynes JF, Yuan ZL, et al. Experimental quantum key distribution beyond the repeaterless secret key capacity. Nat Photon. 2019;13(5):334–8. doi: 10.1038/s41566-019-0377-7

26. Wang R, Yin ZQ, Lu FY, Wang S, Chen W, Zhang CM, et al. Optimized protocol for twin-field quantum key distribution. Commun Phys. 2020;3(1):149. doi: 10.1038/s42005-020-00415-0

27. Currás-Lorenzo G, Wooltorton L, Razavi M. Twin-field quantum key distribution with fully discrete phase randomization. Phys Rev Appl. 2021;15:014016. doi: 10.1103/PhysRevApplied.15.014016

28. Chen JP, Zhang C, Liu Y, Jiang C, Zhang WJ, Han ZY, et al. Twin-field quantum key distribution over a 511 km optical fibre linking two distant metropolitan areas. Nat Photon. 2021;15(8):570–5. doi: 10.1038/s41566-021-00828-5

29. Park CH, Woo MK, Park BK, Kim YS, Baek H, Lee SW, et al. 2xn twin-field quantum key distribution network configuration based on polarization, wavelength, and time division multiplexing. NPJ Quantum Inf. 2022;8:48. doi: 10.1103/PhysRevA.103.012606

30. Teng J, Lu FY, Yin ZQ, Fan-Yuan GJ, Wang R, Wang S, et al. Twin-field quantum key distribution with passive-decoy state. New J Phys. 2020;22(10):103017. doi: 10.1088/1367-2630/abbab7

31. Wang S, Yin ZQ, He DY, Chen W, Wang RQ, Ye P, et al. Twin-field quantum key distribution over 830-km fibre. Nat Photon. 2022;16:154–61. doi: 10.1038/s41566-021-00928-2

32. Kuang R, Barbeau M. Quantum permutation pad for universal quantum-safe cryptography. Quantum Inf Process. 2022;21:211. doi: 10.1007/s11128-022-03557-y

33. Shannon CE. Communication theory of secrecy systems. Bell Syst Tech J. 1949;28(4):656–715. doi: 10.1002/j.1538-7305.1949.tb00928.x

34. Kuang R, Bettenburg N. Shannon perfect secrecy in a discrete hilbert space. In 2020 IEEE International Conference on Quantum Computing and Engineering (QCE). Denver (CO): IEEE; 2020. p. 249–55. doi: 10.1109/QCE49297.2020.00039

35. Lou D, He A, Redding M, Geitz M, Toth R, Döring R, et al. Benchmark performance of digital qkd platform using quantum permutation pad. IEEE Access. 2022;10:107066–76. doi: 10.1109/ACCESS.2022.3212738

36. Kuang R, Lou D, He A, McKenzie C, Redding M. Pseudo quantum random number generator with quantum permutation pad. In 2021 IEEE International Conference on Quantum Computing and Engineering (QCE). Broomfield (CO): IEEE; 2021. p. 359–64. doi: 10.1109/QCE52317.2021.00053

37. Kuang R, Perepechaenko M. Quantum encryption with quantum permutation pad in ibmq systems. EPJ Quantum Technol. 2022;9:26. doi: 10.1140/epjqt/s40507-022-00145-y

38. Perepechaenko M, Kuang R. Quantum encryption of superposition states with quantum permutation pad in IBM quantum computers. EPJ Quantum Technol. 2023;10(1):7. doi: 10.1140/epjqt/s40507-023-00164-3

39. Burge I, Mai MT, Barbeau M. A permutation dispatch circuit design for quantum permutation pad symmetric encryption. In 2024 13th International Conference on Communications, Circuits and Systems (ICCCAS); 2024; Xiamen. p. 35–40. doi: 10.1109/ICCCAS62034.2024.10652827

40. Chancé A. Quantum permutation pad with qiskit runtime. In: Femmam S, Lorenz P, editors. Recent advances in communication networks and embedded systems. Cham: Springer International Publishing; 2024. p. 136–47. ISBN 978-3-031-59619-3.

41. Merkle R, Hellman M. Hiding information and signatures in trapdoor knapsacks. IEEE Trans Inf Theory. 1978;24(5):525–30. doi: 10.1109/TIT.1978.1055927

42. Shamir A. A polynomial time algorithm for breaking the basic merkle-hellman cryptosystem. In 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982); 1982; Chicago (IL). p. 145–52. doi: 10.1109/SFCS.1982.5

43. Qu M, Scott V. The knapsack problem in cryptography. Finite Fields Theory Appl Algorithms. 1994;168:291.

44. Niemi V. A new trapdoor in knapsacks. In Workshop on the Theory and Application of of Cryptographic Techniques. Aarhus: Springer; 1990. p. 405–11.

45. Orton G. A multiple-iterated trapdoor for dense compact knapsacks. In Workshop on the Theory and Application of of Cryptographic Techniques. Berlin, Heidelberg: Springer; 1994. p. 112–30. doi: 10.1007/BFb0053429

46. Wang B, Hu Y. Quadratic compact knapsack public-key cryptosystem. Comput Math Appl. 2010;59(1):194–206. ISSN 0898-1221. doi: 10.1016/j.camwa.2009.08.031.

47. Lagarias JC. Knapsack public key cryptosystems and diophantine approximation. Boston (MA): Springer US; 1984. p. 3–23. ISBN 978-1-4684-4730-9. doi: 10.1007/978-1-4684-4730-9_1

48. Herold G, Meurer A. New attacks for knapsack based cryptosystems. In: Visconti I, De Prisco R, editors. Security and cryptography for networks. Berlin, Heidelberg: Springer; 2012. p. 326–42. ISBN 978-3-642-32928-9.

49. Nguyen P, Stern J. Merkle-Hellman revisited: a cryptanalysis of the qu-vanstone cryptosystem based on group factorizations. In: Kaliski BS, editor. Advances in cryptology — CRYPTO '97. Berlin, Heidelberg: Springer; 1997. p. 198–212. ISBN 978-3-540-69528-8. doi: 10.1007/BFb0052236

50. Kuang R, Perepechaenko M, Barbeau M. A new post-quantum multivariate polynomial public key encapsulation algorithm. Quantum Inf Process. 2022;21:360. doi: 10.1007/s11128-022-03712-5

51. Kuang R, Perepechaenko M, Barbeau M. A new quantum-safe multivariate polynomial public key digital signature algorithm. Sci Rep. 2022;12:13168. doi: 10.1038/s41598-022-15843-x

52. Kuang R, Perepechaenko M. Homomorphic polynomial public key encapsulation over two hidden rings for quantum-safe key encapsulation. Quantum Inf Process. 2023;22:315. doi: 10.1007/s11128-023-04064-4

53. Kuang R, Perepechaenko M, Sayed M, Lou D. Homomorphic polynomial public key with the barrett transformation for digital signature. Acad Quantum. 2024;1. doi: 10.20935/AcadQuant7353

54. Wishart J. Statistical tables for biological agricultural and medical research. Nature. 1939;144:533. doi: 10.1038/144533a0

55. Amil A, Gupta S. Cryptanalysis of quantum permutation pad. ArXiv. 2023:abs/2304.11081. doi: 10.48550/arXiv.2304.11081

56. Barbeau M. Quantum data communication protection with the quantum permutation pad block cipher in counter mode and clifford operators. F1000Res. 2023;12:1123. doi: 10.12688/f1000research.140027.1

57. Facchini S, Perdrix S. Quantum circuits for the unitary permutation problem. In: Jain R, Jain S, Stephan F, editors. Theory and applications of models of computation. Cham: Springer International Publishing; 2015. p. 324–31. ISBN 978-3-319-17142-5. doi: 10.1007/978-3-319-17142-5_28

58. Soeken M, Mozafari F, Schmitt B, De Micheli G. Compiling permutations for superconducting qpus. In 2019 Design, Automation and Test in Europe Conference and Exhibition (DATE); 2019; Florence. p. 1349–54. doi: 10.23919/DATE.2019.8715275

59. Liu J, Ren Y, Cao Y, Sun H, Chen L. Realization of permutation groups by quantum circuit; 2024 [cited 2024 Nov 27]. Available from: https://arxiv.org/abs/2406.01350

60. Yu A. Quantum complexity of permutations; 2022 [cited 2024 Nov 27]. Available from: https://arxiv.org/abs/2207.14102

61. Shor PW. Algorithms for quantum computation: discrete logarithms and factoring. In Proceedings 35th Annual Symposium on Foundations of Computer Science. IEEE; 1994; Santa Fe (NM). p. 124–34. doi: 10.1109/SFCS.1994.365700

62. Odlyzko AM. The rise and fall of knapsack cryptosystems; 1998 [cited 2024 Nov 27]. Available from: https://www-users.cse.umn.edu/~odlyzko/doc/arch/knapsack.survey.pdf

63. Marsaglia G. Xorshift rngs. J Stat Softw. 2003;8(14):1. doi: 10.18637/jss.v008.i14.

64. Vigna S. An experimental exploration of marsaglia's xorshift generators, scrambled. ACM Trans Math Softw. 2016;42(4). doi: 10.1145/2845077

65. Blackman D, Vigna S. Scrambled linear pseudorandom number generators; 2022 [cited 2024 Nov 27]. Available from: https://arxiv.org/abs/1805.01407

66. Hurley-Smith D, Hernandez-Castro J. Quam bene non quantum: bias in a family of quantum random number generators. IACR Cryptol ePrint Arch; 2017:842. Available from: https://api.semanticscholar.org/CorpusID:3691425

67. Quantum key distribution: protocol and data format of rest-based key delivery api; 2019 [cited 2024 May 30]. Available from: https://cdn.standards.iteh.ai/samples/53603/549fd020d0654b3bb9ad052fec3c0e9a/ETSI-GS-QKD-014-V1-1-1-2019-02-.pdf

68. qstream 200 plus quantum entropy appliance. [cited 2024 May 30]. Available from: https://info.quintessencelabs.com/hubfs/QLabs-qStream-product-sheet.pdf

69. Quantis qrng pcie. 2024 [cited 2024 May 30]. Available from: https://www.idquantique.com/random-number-generation/products/quantis-qrng-pcie/

70. Ding J, Yang BY. Multivariate public key cryptography. Berlin, Heidelberg: Springer; 2009. p. 193–241. doi: 10.1007/978-3-540-88702-7_6

71. Piotr Zawadzki. A chosen-plaintext attack on quantum permutation pad. Quantum Inf Process. 2024;23(3):73. doi: 10.1007/s11128-024-04278-0

72. Hwang MS, Lee CC, Tzeng F. A new knapsack public-key cryptosystem based on permutation combination algorithm. World Acad Sci Eng Technol Int J Comput Electr Autom Control Inf Eng. 2009;3:2291–6. Available from: https://api.semanticscholar.org/CorpusID:15081482

73. Murakami Y, Katayanagi K, Kasahara M. A new class of cryptosystems based on chinese remainder theorem. In 2008 International Symposium on Information Theory and Its Applications; 2008; Auckland. p. 1–6. doi: 10.1109/ISITA.2008.4895587

74. Su S, Lü S. A public key cryptosystem based on three new provable problems. Theor Comput Sci. 2012;426–7:91–117. doi: 10.1016/j.tcs.2011.12.011

75. Bi J, Han L, Meng X. Cryptanalysis of two knapsack public-key cryptosystems. IACR Cryptol ePrint Arch. 2009:537. [cited 2024 Nov 27]. Available from: https://eprint.iacr.org/2009/537

76. Peng L, Hu L, Xu J, Xie Y, Zuo J. Analysis of two knapsack public key cryptosystems. IET Commun. 2013;7(15):1638–43. doi: 10.1049/iet-com.2013.0180

77. Faugére JC. A new efficient algorithm for computing gröbner bases (f4). J Pure Appl Algebra. 1999;139(1):61–88. doi: 10.1016/S0022-4049(99)00005-5

78. Joux A, Vitse V. A Variant of the F4 algorithm. In: Kiayias A, editor. Topics in cryptology – CT-RSA 2011. Lecture notes in computer science. Vol 6558. Berlin, Heidelberg: Springer; 2011. doi: 10.1007/978-3-642-19074-2_23

79. Yang BY, Chen JM. All in the xl family: theory and practice. In: Park CS, Chee S, editors. Information security and cryptology – ICISC 2004. Berlin, Heidelberg: Springer; 2005. p. 67–86. ISBN 978-3-540-32083-8.

80. Beullens W. Improved cryptanalysis of uov and rainbow. In Advances in Cryptology–EUROCRYPT 2021. Volume 12697 of Lecture Notes in Computer Science. Cham: Springer; 2021. p. 348–73.

81. Nakamura S, Ikematsu Y, Takagi T. Recent progress in the security evaluation of multivariate public-key cryptography. IET Inf Secur. 2023;17(2):210–26. doi: 10.1049/ise2.12092

82. Hashimoto Y. Recent developments in multivariate public key cryptosystems. In: Takagi T, Wakayama M, Tanaka K, Kunihiro N, Kimoto K, Ikematsu Y, editors. International Symposium on Mathematics, Quantum Theory, and Cryptography. Mathematics for Industry. Vol 33. Singapore: Springer; 2021. doi: 10.1007/978-981-15-5191-8_16

83. Li Z, Wu B, Lin D. Algebraic-differential attacks on a family of arithmetization-oriented symmetric ciphers. J Syst Sci Complex. 2023;36:2681–702. doi: 10.1007/s11424-023-1511-7

84. Wang M, Wang X, Hui L. Differential-algebraic cryptanalysis of reduced-round of serpent 256. Sci China Inf Sci. 2010;53:546–56. doi: 10.1007/s11432-010-0048-2

85. Albrecht MR, Cid C, Grassi L, Khovratovich D, Lüftenegger R, Rechberger C, et al. Algebraic cryptanalysis of STARK-friendly designs: application to MARVELLOUS and MiMC. In: Galbraith S, Moriai S, editors. Advances in cryptology – ASIACRYPT 2019. Lecture notes in computer science. Vol. 11923. Cham: Springer; 2019. doi: 10.1007/978-3-030-34618-8_13

86. Bardet M, Briaud P, Bros M, Gaborit P, Tillich JP. Revisiting algebraic attacks on MinRank and on the rank decoding problem. Cryptol ePrint Arch. 2022;2022/1031. [cited 2024 Nov 27]. Available from: https://eprint.iacr.org/2022/1031.

87. Bardet M, Pierre Briaud, Maxime Bros, Philippe Gaborit, Vincent Neiger, Olivier Ruatta, et al. An Algebraic attack on rank metric code-based cryptosystems. In: Canteaut A, Ishai Y, editors. Advances in cryptology – EUROCRYPT 2020. Lecture notes in computer science. Vol. 12107. Cham: Springer; 2020. doi: 10.1007/978-3-030-45727-3_3

88. Bardet M, Bros M, Cabarcas D, Gaborit P, Perlner R, Smith-Tone D, et al. Improvements of algebraic attacks for solving the rank decoding and MinRank problems. In: Moriai S, Wang H, editors. Advances in cryptology – ASIACRYPT 2020. Lecture notes in computer science. Vol. 12491. Cham: Springer; 2020. https://doi.org/10.1007/978-3-030-64837-4_17