ACADEMIA QUANTUM

Research Article
Published: 2025-07-31
https://doi.org/10.20935/AcadQuant7841

Quantum cryptographic dynamics: modeling cryptosystems via entropy operators

Randy Kuang[1,*]

Academic Editor: Guilherme Temporao

Abstract

This paper introduces Quantum Cryptographic Dynamics (QCD), a novel theoretical framework that models cryptographic processes through the lens of entropy injection and ejection. Drawing inspiration from classical mechanics, QCD establishes three fundamental laws: the Entropy Inertial Law (conservation of entropy in isolated systems), the Entropy Evolution Law (transformation of entropy via injection operators), and the Entropy Redistribution Law (reversibility through ejection operators). Applying these principles, we provide a unified, entropy-centric interpretation of classical and quantum cryptographic schemes, including symmetric-key systems, public-key protocols, and post-quantum cryptography (PQC) algorithms such as Learning With Errors (LWE), Kyber, and Homomorphic Polynomial Public Key (HPPK). By shifting the focus from computational hardness assumptions to the fundamental dynamics of entropy manipulation, QCD offers new insights into the security foundations of these cryptographic primitives. Furthermore, we reinterpret the Quantum Permutation Pad Random Number Generator (QPP-RNG) within the QCD framework. QPP-RNG is modeled as an entropy-driven process that harnesses system jitter to generate unpredictable random numbers, which in turn fuel PQC schemes like Kyber and HPPK for quantum-secure key establishment, forming a self-sustained quantum-secure eco-cryptosystem. This framework provides a rigorous approach to security analysis, unifying cryptographic security models across classical, quantum, and post-quantum domains. QCD establishes a robust foundation for designing quantum-resistant cryptographic primitives and assessing the fundamental security properties of cryptographic systems.

Keywords: *quantum cryptography; quantum cryptographic dynamics; entropy injection; entropy ejection; learning with errors; Kyber; homomorphic polynomial public key; post-quantum cryptography; cryptographic primitives; QPP; QPP-RNG*

Citation: Kuang R. Quantum cryptographic dynamics: modeling cryptosystems via entropy operators. *Academia Quantum* 2025;2. https://doi.org/10.20935/AcadQuant7841

1. Introduction

Cryptography, traditionally rooted in Boolean algebra and number theory, faces a paradigm shift with the advent of quantum computing. This paper introduces *Quantum Cryptographic Dynamics (QCD)*, a novel framework within information theory that employs linear algebra to describe the dynamic evolution of information states under cryptographic transformations.

Classical systems like RSA [1], AES [2], and ECC [3], which rely on static mathematical structures and computational hardness assumptions (e.g., integer factorization, discrete logarithms), are vulnerable to quantum algorithms such as Shor's algorithm [4]. Quantum cryptographic protocols, such as Quantum Key Distribution (QKD) [5, 6], offer security independent of these assumptions by exploiting quantum mechanics. However, a unified theoretical framework encompassing both classical and quantum symmetric and asymmetric cryptography has been absent.

QCD addresses this gap by modeling cryptographic transformations as linear operators acting on information states, analogous to quantum mechanics. Instead of static mappings, QCD views encryption and decryption as dynamic processes of entropy injection and ejection:

- $|m\rangle$: plaintext (message state)
- $|c\rangle$: ciphertext (transformed state)
- $\hat{p}_{in}$: entropy injection operator, where $\hat{p}_{in}|m\rangle = |c\rangle$
- $\hat{p}_e$: entropy ejection operator, where $\hat{p}_e|c\rangle = |m\rangle$

These operators are governed by three fundamental laws describing entropy invariance, evolution, and redistribution.

1.1. Motivation for quantum cryptographic dynamics (QCD)

1.1.1. Unified formalism for classical, quantum, and asymmetric cryptography

QCD unifies classical (Boolean algebra, discrete probability) and quantum (Hilbert space, quantum operators) cryptography within a common linear algebraic framework in $\mathbb{C}^{2^n}$. Cryptographic transformations are expressed as pairs of entropy operators. Furthermore, QCD differentiates asymmetric and symmetric schemes based on whether the entropy injection operator is public or secret.

[1]Quantropi Inc., 1545 Carling Ave., Suite 620, Ottawa, ON, Canada.
*email: randy.kuang@quantropi.com

1.1.2. Entropy-based security via operator robustness

Instead of relying purely on computational hardness assumptions, QCD characterizes security through entropy dynamics by constructing paired entropy injection and ejection operators. The security of these operators can still be connected to well-known computational hardness problems. For instance, RSA's injection operator $\hat{p}_{in} = \square^e \bmod N$ depends on the difficulty of factoring N. Thus, QCD reframes security as the concrete robustness of operator pairs against both classical and quantum attacks.

1.1.3. Systematic framework for cryptographic analysis and design

QCD offers a principled, axiomatic approach to cryptographic analysis and design, analogous to classical mechanics. Its three fundamental laws—entropy invariance, evolution, and redistribution—provide a systematic framework that unifies classical and quantum cryptography, while distinguishing symmetric and asymmetric schemes through the publicity of the entropy injection operator.

1.1.4. Information systems as physical systems governed by entropic laws

QCD models information systems similarly to physical systems, where entropy is treated as a measurable, dynamic quantity transformed by external operations. Just as physical systems are governed by dynamical laws (e.g., Newton's laws or quantum evolution), QCD posits three entropic laws that describe how information entropy evolves under cryptographic processes. This perspective allows a unified, physics-inspired formulation of cryptography within Hilbert space and linear operators.

1.2. Structure of this paper

The remainder of this paper is organized as follows. **Section 2** presents the three fundamental laws of QCD, drawing parallels to classical physics. **Section 3** examines major cryptographic schemes through the QCD perspective. Finally, **Section 4** outlines future research directions.

By establishing Quantum Cryptographic Dynamics, we offer a unified, entropy-based security model that integrates classical and quantum systems, enabling rigorous analysis of cryptographic transformations.

2. Methods

Quantum Cryptographic Dynamics (QCD) is a generalized theoretical framework that models cryptographic processes using the formalism of linear algebra in Hilbert space. It defines entropy manipulation in terms of injection and ejection operators acting on information states, governed by three dynamical laws. While QCD applies to both classical and quantum cryptographic systems, its formulation using Dirac notation and operator-based dynamics also supports direct implementation in physical quantum computing systems.

In particular, entropy operators in QCD can be instantiated as unitary transformations on quantum states, making the framework suitable for quantum circuit realization. Thus, the quantum in QCD refers not only to its mathematical expression but also to its compatibility with quantum cryptographic protocols such as QKD and quantum-secure key exchange. The three laws of QCD provide a unified model for reasoning about cryptographic systems across classical and quantum domains.

First Law—Entropy Inertia

The entropy of an information system remains invariant unless influenced by an external cryptographic operation.

Interpretation: Analogous to Newton's first law of motion, this principle establishes entropy—a measure of uncertainty or randomness—as an inherent property of information systems. A system in equilibrium maintains its entropy profile; plaintext retains its structural predictability, while ciphertext preserves its encoded randomness. Formally, Shannon entropy H is defined as [7]

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x), \tag{1}$$

where $p(x)$ denotes the probability of symbol x in the alphabet $\mathcal{X}$.

Empirical Foundations: Seminal studies reveal varying entropy measurements across languages:

- English text: 2.62 bits/letter (frequency analysis) [7].
- Human prediction: 0.6–1.3 bits/character [8].
- Modern estimates: 1.75 bits/character [9].
- Chinese logograms: ~9.56 bits/character [10].

Manifestations:

- **Static Data Preservation:** Isolated plaintext files maintain fixed entropy profiles; for example, an unmodified document retains its original information content indefinitely, absent of compression or encryption.
- **Quantum State Invariance:** Qubits in pure states $|\psi\rangle$ exhibit zero von Neumann entropy ($S = -\mathrm{tr}(\rho \ln \rho)$), persisting until quantum operations induce mixed states or decoherence.
- **Passive Communication:** Unencrypted transmissions preserve the source entropy. Entropy amplification requires active cryptographic processes (e.g., AES, OTP).

This entropy inertia principle establishes the axiomatic foundation for information equilibrium in QCD, showing a system's inherent resistance to entropy change in the absence of external operations.

Second Law—Entropy Evolution

An information system's entropy transforms under an entropy injection operator $\hat{p}_{in} \in \mathbf{P}$:

$$\hat{p}_{in}|s\rangle = |t\rangle, \tag{2}$$

where $|s\rangle$ and $|t\rangle$ represent the pre- and post-operation states of the system.

Interpretation: The dimension N of the entropy space $\mathbf{P}$ determines the operator's entropy:

$$e_p = \log_2 N, \tag{3}$$

which quantifies the uncertainty of operator selection.

Classical vs. Quantum Spaces:

- **Classical:** An n-bit key space (with 2^n keys) in Boolean algebra yields n bits of entropy.
- **Quantum:** An n-bit key space in linear algebra or permutation space (with $2^n!$ operators) yields

$$e_p = \log_2(2^n!) \approx 2^n(n\ln 2 - 1) \quad \text{(using Stirling's approximation [11]).} \tag{4}$$

Cryptographic Parallel: Analogous to Newton's second law ($F/m = a$), the entropy operator $\hat{p}$ acts as a cryptographic force, shielding plaintext $|s\rangle$ from observation. Just as force can obscure the measurement of mass, $\hat{p}$ prevents the ciphertext $|t\rangle$ from revealing the plaintext without knowledge of the operator.

Third Law—Entropy Redistribution

For every entropy injection operator $\hat{p}_{in}$, there exists an ejection operator $\hat{p}_e$ such that

$$\hat{p}_e\hat{p}_{in} = \mathbb{I}, \tag{5}$$

ensuring reversibility and conservation of entropy.

Interpretation: This law requires entropy injection and ejection operators to be reversible—a property inherent in permutations and quantum unitary operations. It unifies classical and quantum cryptography by modeling both as operator dynamics in permutation or linear spaces.

Complementarity with the Second Law: Together, these ensure

$$\hat{p}_e(\hat{p}_{in}|s\rangle) = |s\rangle, \tag{6}$$

guaranteeing that information is recoverable while remaining secure during transmission.

Security Implications: Cryptographic strength depends on designing robust entropy operators that withstand classical-, quantum-, and machine-learning-based attacks. The operator design remains central to achieving secure systems.

The next section analyzes major cryptographic paradigms through the QCD perspective, highlighting new insights into their security and quantum resilience.

3. Results and Discussion

The framework of Quantum Cryptographic Dynamics (QCD) provides a novel lens for analyzing both classical and quantum cryptographic schemes. By understanding encryption and decryption as entropy injection and ejection processes, we can categorize cryptographic mechanisms based on their entropy operator structures: secret or public injection operators. This section examines major cryptographic paradigms in light of the three QCD laws.

3.1. Symmetric-key cryptography: secret injection operators

Symmetric-key cryptography encrypts a plaintext message into ciphertext using a secret key and decrypts it back using the same key. This process can be described mathematically as the application of an entropy injection operator ($\hat{p}_i$) to a message state $|s\rangle$:

$$\hat{p}_i|s\rangle = |t\rangle, \tag{7}$$

where $|s\rangle$ represents the plaintext and $|t\rangle$ represents the ciphertext. According to the third law of QCD, an entropy ejection operator

$$\hat{p}_e\hat{p}_i = \mathbb{I}, \tag{8}$$

ensuring that entropy is injected during encryption and completely extracted during decryption.

3.1.1. OTP

A notable example is the One-Time Pad (OTP) scheme, as introduced by Shannon [7]. In this framework, the encryption and decryption operators can be represented as

$$\hat{p}_i = \hat{p}_e = k \oplus \square, \tag{9}$$

where $\square$ denotes the information state. Thus, the encryption process follows

$$\hat{p}_i|s\rangle = |k \oplus s\rangle = |t\rangle, \tag{10}$$

and the corresponding decryption process is

$$\hat{p}_e|t\rangle = |k \oplus t\rangle = |k \oplus k \oplus s\rangle = |s\rangle. \tag{11}$$

This follows from the fundamental properties of the XOR operation, where $k \oplus k = 0$ and $0 \oplus s = s$.

Shannon's proof demonstrated that OTP provides *perfect secrecy*, meaning that the ciphertext reveals no information about the plaintext, provided that the key k is uniformly random, has the same length as the message, and is used only once. This makes OTP a theoretically unbreakable encryption scheme, though its practical implementation is constrained by key management challenges.

3.1.2. AES

To overcome the constraint of single-use keys in the One-Time Pad (OTP), block ciphers such as DES and AES were developed. In this section, we illustrate AES encryption and decryption using a permutation framework with quantum-inspired (ket) notation to emphasize the abstract and layered structure of the cipher.

AES is a block cipher with a fixed block size of 128 bits. A plaintext block $|m\rangle$ is transformed into a ciphertext block $|c\rangle$ by a permutation operator $\hat{p}_{AES}$ chosen from the space of all 128-bit permutations:

$$\hat{p}_{AES}|m\rangle = |c\rangle. \tag{12}$$

In practice, AES encryption is executed over multiple rounds, each involving a permutation operator $\hat{p}_i$ that transforms the state from one intermediate stage to the next:

$$\hat{p}_{AES} = \hat{p}_M\,\hat{p}_{M-1}\cdots\hat{p}_2\,\hat{p}_1, \tag{13}$$

so that the encryption (or entropy injection) process can be expressed as

$$\begin{aligned}\hat{p}_{AES}|m\rangle &= \hat{p}_M \hat{p}_{M-1} \cdots \hat{p}_1 |m\rangle \\ &= \hat{p}_M \hat{p}_{M-1} \cdots \hat{p}_2 |c_1\rangle \\ &\vdots \\ &= |c\rangle,\end{aligned} \tag{14}$$

where M denotes the number of rounds. For instance, AES-128 uses $M = 10$ rounds, while AES-256 uses $M = 14$ rounds.

Decryption (or entropy ejection) is the reverse process, employing the inverse permutation operators:

$$\hat{p}_{AES}^{-1} = \hat{p}_1^{-1} \hat{p}_2^{-1} \cdots \hat{p}_M^{-1}, \tag{15}$$

Therefore, applying $\hat{p}_{AES}^{-1}$ to the ciphertext recovers the original plaintext:

$$\begin{aligned}\hat{p}_{AES}^{-1}|c\rangle &= \hat{p}_1^{-1} \hat{p}_2^{-1} \cdots \hat{p}_M^{-1} |c\rangle \\ &= \hat{p}_1^{-1} \hat{p}_2^{-1} \cdots \hat{p}_{M-1}^{-1} |c_{M-1}\rangle \\ &\vdots \\ &= \hat{p}_1^{-1} |c_1\rangle \\ &= |m\rangle.\end{aligned} \tag{16}$$

Within each round, the permutation operator $\hat{p}_i$ is implemented through four distinct operations:

1. **SubBytes ($\hat{p}_{Sub}$):** This step can be viewed as a simplified 128-bit permutation constructed as the direct product of 16 identical 8-bit permutations. In practice, a chosen 256×256 permutation matrix is applied independently to each of the 16 individual 8-bit segments of the 16-byte block. Although the operation is performed on each byte separately, the overall effect is equivalent to applying a single permutation to the entire 128-bit block, thereby introducing the essential nonlinearity (confusion) into the cipher.
2. **ShiftRows ($\hat{p}_{SR}$):** This operation rearranges the positions of the bytes within the 16-byte block. By cyclically shifting the rows, it ensures that the bytes are diffused across different columns in subsequent operations, thereby enhancing overall diffusion.
3. **MixColumns ($\hat{p}_{MC}$):** Here, a reversible linear transformation is applied using a 16×16 block diagonal matrix composed of four 4×4 circulant matrices. This step further mixes the bytes within each column, reinforcing the diffusion properties of the cipher.
4. **AddRoundKey ($\hat{p}_{ARK}$):** In this final step, a 16-byte round key is XORed with the output of the MixColumns operation. Although XOR is a bitwise operation, it constitutes a bijective transformation over the finite field, thereby integrating seamlessly into the permutation framework.

Thus, each round permutation operator can be expressed as a composition of these sub-operations:

$$\hat{p}_i = \hat{p}_{Sub}\, \hat{p}_{SR}\, \hat{p}_{MC}\, \hat{p}_{ARK}. \tag{17}$$

It is important to note that while $\hat{p}_{AES}$ is mathematically a bijection over the space of 128-bit blocks, it is not an arbitrary permutation. Rather, it is a carefully designed composite permutation, constructed from well-defined arithmetic operations, to achieve both confusion and diffusion, which are essential for ensuring the cipher's security.

In the full permutation space of 128-bit blocks there are $(2^{128})!$ possible bijective permutations, yet the AES-128 key space realizes only 2^{128} unique permutations—one for each key. Despite this, AES is engineered to approximate the behavior of a random permutation drawn from the full space, and its excellent performance in both bit-level and byte-level randomness testing reflects the remarkable design considerations for achieving robust confusion and diffusion.

3.1.3. Quantum permutation pad (QPP)

Unlike AES, which employs mathematically structured permutation operators drawn from the full 128-bit permutation space and relies on multiple rounds of layered operations to achieve randomness and security, Quantum Permutation Pad (QPP) cryptography utilizes unstructured (or structureless) permutation matrices in a parallel configuration. This approach, introduced by Kuang and Barbeau [12] in 2022, integrates three key sub-processes: pre-randomization, random dispatch, and QPP encryption.

In the *pre-randomization* stage, the plaintext is XORed with a pseudorandom value k generated by a PRNG seeded with the shared secret. This operation is expressed as

$$\hat{p}_{k\oplus}|m\rangle = |k \oplus m\rangle = |m'\rangle,$$

where $|k\rangle$ represents the PRNG output and $\hat{p}_{k\oplus}$ denotes the XOR permutation operator. This initial entropy injection obscures the plaintext and prepares it for further processing.

The *random dispatch* stage then uses the same PRNG to control the distribution of the pre-randomized data across multiple parallel channels. In this process, the pre-randomized block is partitioned and routed according to random dispatch operators, thereby enhancing diffusion by spreading the entropy over distinct channels.

The final stage, *QPP encryption*, applies a set of parallel permutation operators generated via the Fisher–Yates shuffling algorithm [13]. These unstructured permutation matrices are seeded by the shared secret and organized into a QPP pad containing M independent permutations. The overall uncertainty in an n-bit permutation space is enormous, since there are

$$2^n!$$

possible bijections, corresponding to a Shannon entropy of

$$e = \log_2(2^n!)$$

bits [11]. In practice, specifying an arbitrary permutation in this space would require a key length of approximately $n \times 2^n$ bits. For example, an 8-bit permutation space exhibits an uncertainty of roughly 1684 bits, meaning that choosing a permutation would require on the order of a 2048-bit key. For a QPP pad with M permutations, the internal uncertainty is given by

$$2^n \times M \times \left(2^n!\right)^M,$$

where the factor 2^n arises from the pre-randomization, M represents the random dispatch across M channels, and $(2^n!)^M$ reflects

the combined uncertainty of M parallel permutations. Consequently, QPP cryptography offers an overall uncertainty of $(2^n!)^M$, which is exponentially larger than that of classical schemes such as AES.

QPP encryption (or entropy injection) can be concisely expressed as

$$\hat{p}_{QPP} = \hat{p}_{(i=k \bmod M)}\, \hat{p}_{(k\oplus)}, \tag{18}$$

where k is an integer generated by the seeded PRNG, $\hat{p}_i$ denotes the ith permutation operator from the QPP pad (with $i = k \bmod M$), and $\hat{p}_{(k\oplus)}$ represents the XOR permutation based on an n-bit key k from the PRNG. The encryption process is then described by

$$\hat{p}_{QPP}|m\rangle = \hat{p}_{(i=k \bmod M)}\, \hat{p}_{(k\oplus)}|m\rangle = \hat{p}_{(i=k \bmod M)}|k \oplus m\rangle = |c\rangle. \tag{19}$$

For decryption (or entropy ejection), the inverse operations are applied in reverse order using the same PRNG-derived keys:

$$\hat{p}^{-1}_{QPP} = \hat{p}^{-1}_{(k\oplus)}\, \hat{p}^{-1}_{(i=k \bmod M)}. \tag{20}$$

Thus, for a given ciphertext $|c\rangle$, decryption proceeds as

$$\hat{p}^{-1}_{QPP}|c\rangle = \hat{p}^{-1}_{(k\oplus)}\, \hat{p}^{-1}_{(i=k \bmod M)}|c\rangle = \hat{p}^{-1}_{(k\oplus)}|k \oplus m\rangle = |m\rangle. \tag{21}$$

In summary, QPP cryptography leverages unstructured, parallel permutation matrices controlled by a shared secret to achieve high entropy injection and rapid performance. This method is particularly well suited for encrypting small blocks of information (such as 4-bit or 8-bit segments), while also exhibiting excellent cipher randomness.

3.2. Classical public key cryptography: public injection operators

The following descriptions of RSA and Diffie–Hellman protocols follow standard treatments in classical cryptography (see, e.g., [14–17]).

Unlike symmetric cryptography, where both the entropy injection (encryption) and ejection (decryption) operators remain secret, public key cryptography employs a public entropy injection operator to establish secure key sharing. One of the earliest and most well-known public key schemes is RSA [17]. In RSA, the entropy injection operator (or permutation operator) is defined as

$$\hat{p}_{RSA} = \square^e \bmod N, \tag{22}$$

where e is the public exponent and N is a modulus formed by the product of two large secret primes. The corresponding secret ejection operator (or reverse permutation operator) is given by

$$\hat{p}^{-1}_{RSA} = \square^d \bmod N, \tag{23}$$

with d being the secret exponent satisfying

$$e \times d \equiv 1 \pmod{\varphi(N)},$$

and $\square$ denoting the input message or cipher. For a chosen secret $s < N$, the encryption (entropy injection) is performed as

$$\hat{p}_{RSA}|s\rangle = |s^e \bmod N\rangle = |c\rangle, \tag{24}$$

and the decryption (entropy ejection) is carried out via

$$\hat{p}^{-1}_{RSA}|c\rangle = |c^d \bmod N\rangle = |s^{e\times d \bmod \varphi(N)} \bmod N\rangle = |s\rangle. \tag{25}$$

The security of RSA in classical computing relies on the computational difficulty of deriving the secret ejection operator $\hat{p}^{-1}_{RSA}$ from the publicly known injection operator $\hat{p}_{RSA}$.

Another widely used public key scheme is Diffie–Hellman (DH) cryptography [16], which operates over a prime finite field $\mathbb{F}_p$ with a public generator g. In this context, the entropy injection operator is defined as

$$\hat{p}_{DH}(g) = g^{\square} \bmod p, \tag{26}$$

where $\square$ represents a chosen secret from $\mathbb{F}_p$. In the DH protocol, Alice selects a secret a and computes

$$\hat{p}_{DH}(g)|a\rangle = |g^a \bmod p\rangle = |A\rangle, \tag{27}$$

which she then sends to Bob. Similarly, Bob chooses his secret b and computes

$$\hat{p}_{DH}(g)|b\rangle = |g^b \bmod p\rangle = |B\rangle, \tag{28}$$

which he sends to Alice. Rather than directly decrypting B or A, both parties perform an additional entropy injection using the other's public value. Specifically, Alice computes

$$\hat{p}_{DH}(B)|a\rangle = |B^a \bmod p\rangle = |g^{ab} \bmod p\rangle = |k\rangle, \tag{29}$$

and Bob computes

$$\hat{p}_{DH}(A)|b\rangle = |A^b \bmod p\rangle = |g^{ab} \bmod p\rangle = |k\rangle. \tag{30}$$

Thus, they arrive at a shared secret

$$k = g^{ab} \bmod p.$$

It is important to note that classical public key cryptography schemes such as RSA and Diffie–Hellman are vulnerable to quantum computing attacks. Quantum algorithms, most notably Shor's algorithm, can efficiently factor large integers and compute discrete logarithms, thereby undermining the security assumptions underlying these schemes. As quantum computing technology advances, there is a growing need to develop post-quantum cryptographic algorithms that remain secure in the presence of quantum adversaries. Such quantum-resistant schemes do not rely on traditional number-theoretic assumptions and offer a promising path toward future-proof secure communications.

3.3. Quantum cryptography

Quantum Key Distribution (QKD) schemes, such as BB84 [18], exploit quantum mechanical principles to establish secure keys between communicating parties. In these protocols, two secret strings are involved; one string, denoted as s_1, is used for key exchange or plaintext $\{|m_i\rangle\}$, while the other, s_2, determines the basis for encoding $\{\ |m_i\rangle\ \}$ or selects the entropy injection operator $\{\ \hat{p}_i\ \}$. Kuang and Barbeau [12] demonstrated that QKD can be interpreted as a one-time pad (OTP) implemented with qubits, where a post-sharing key is established through a public announcement process.

The permutation space for a single qubit comprises two operators: the identity operator $\hat{p}_1$ and the XOR permutation $\hat{p}_2$. Within the framework of Quantum Cryptographic Dynamics (QCD), the encoding process in QKD can be described as

$$\hat{p}_i|m_i\rangle = |c_i\rangle, \tag{31}$$

where $\hat{p}_i = \hat{p}_1$ if the corresponding i-th bit is zero, and $\hat{p}_i = \hat{p}_2$ if the i-th bit is one. Here, $|c_i\rangle$ represents the cipher qubit that embodies the encrypted bit. To extract the original bit m_i, the receiver must measure $|c_i\rangle$ in the computational basis using the correct entropy ejection operator, $\hat{p}_i^{-1}$.

Due to the no-cloning theorem, an adversary cannot perfectly replicate the qubit $|c_i\rangle$. Any attempt to measure or copy the qubit introduces disturbances that contribute to the Quantum Bit Error Rate (QBER), thereby alerting the trusted receiver to the presence of eavesdropping. However, because the trusted receiver does not initially know the sender's chosen entropy injection operator $\{\hat{p}_i\}$, they must measure the incoming qubits using a randomly selected entropy ejection operator $\{\hat{p'}_i\}$ from the permutation space. This measurement yields a set of outcomes $\{m_i'\}$ described by

$$\hat{p'}_i|c_i\rangle = |m_i'\rangle. \tag{32}$$

Subsequently, a public announcement phase is used to reconcile the measurement bases, allowing the communicating parties to discard the bits obtained using incorrect entropy ejection operators. The remaining bits form the raw key, effectively serving as the decryption key in accordance with the third law of QCD.

From a system security perspective, it is important to note that QKD is not a purely quantum cryptographic method, as it relies on a classical channel to synchronize the entropy injection and ejection operators. The incorporation of a classical channel necessitates a trusted relationship between the parties to mitigate the risk of Man-in-the-Middle (MITM) attacks.

3.4. Post-quantum cryptography

3.4.1. Learning with errors (LWE) in a QCD framework

Post-quantum cryptographic schemes are designed to remain secure even against quantum adversaries. A prominent example is cryptography based on the Learning With Errors (LWE) problem. In a QCD-inspired view, encryption is seen as an entropy injection process (in line with the QCD second law), while decryption represents an entropy ejection process (in accord with the QCD third law).

In an LWE-based scheme [19], a public key is generated by choosing a random matrix

$$A \in \mathbb{Z}_q^{m\times n}$$

and a secret vector

$$s \in \mathbb{Z}_q^n,$$

together with a small error vector

$$e \in \mathbb{Z}_q^m,$$

which is also sampled. The entropy injection operator in key generation is defined as

$$\hat{p}_{\mathrm{LWE}} = A\,\square + e,$$

where the symbol $\square$ denotes a secret vector. In this framework, Alice performs an entropy injection with her chosen secret s:

$$\hat{p}_{\mathrm{LWE}}|s\rangle = |A\,s + e\rangle = |b\rangle,$$

and transmits the pair (A, b) to Bob.

For encryption, Bob selects a random vector r (which serves as his ephemeral entropy source) and applies his entropy injection operator similarly to Alice's but transposed secret vector without the error vector:

$$\hat{p}'_{\mathrm{LWE}}|r\rangle = |r^T A\rangle = |u\rangle.$$

This allows Alice to establish a runtime decryption key or entropy ejection operator. Bob then establishes a runtime entropy injection key

$$\hat{k} = r^T b + \square,$$

which is used to mask a scaled version of the plaintext μ. In particular, the encryption operator acts as

$$\hat{k}|\mu\rangle = |r^T b + \left\lfloor \frac{q}{2} \right\rfloor \mu \bmod q\rangle = |v\rangle.$$

The ciphertext is given by the pair (u, v).

Decryption (entropy ejection) is carried out by Alice using her secret key s. In a QCD analogy, she employs an ejection operator, which is effectively the inverse of the encryption process:

$$\hat{k}^{-1}|v\rangle = |v - u\cdot s\rangle = |\delta\rangle.$$

Finally, by comparing δ with $\lfloor q/2 \rfloor$, Alice recovers the plaintext μ.

However, due to the presence of the error term, the relationship between the runtime entropy injection operator and the entropy ejection operator is not perfectly unitary:

$$\hat{k}\hat{k}^{-1} \neq 1. \tag{33}$$

This discrepancy introduces a small probability of decryption failure, as the accumulated noise may occasionally shift δ outside the correct decision boundary when mapping back to the plaintext message.

In summary, within the QCD framework, the following occurs:

- The entropy injection operator (QCD second law) injects randomness and noise into the secret state to produce a public key;
- The encryption process uses a runtime key to mask the scaled message, ensuring that the ciphertext reflects both the injected entropy and the underlying noise;
- The decryption process (QCD third law) employs the secret to eject the injected entropy and recover the original message, though with a small probability of failure due to noise accumulation.

The security of LWE-based schemes relies on the computational hardness of solving the LWE problem, which remains difficult even for quantum adversaries based on the hardness of lattice problems.

This interpretation of LWE within the QCD framework not only provides a structured view of entropy management in encryption but also underscores why LWE remains one of the leading candidates for post-quantum cryptography.

Kyber [20], as a lattice-based post-quantum cryptographic scheme, can be interpreted within the QCD framework as an entropy injection and ejection process. Key generation injects entropy through a public module-LWE matrix and noise, forming

a public key that conceals the secret. Encryption introduces additional entropy via an ephemeral key, ensuring security against adversaries. Decryption acts as an entropy ejection process, attempting to recover the plaintext by reversing the transformations. However, due to the presence of noise, the runtime entropy injection and ejection operators are not perfectly reversible, leading to a small probability of decryption failure. This aligns with the fundamental QCD principle that entropy injection increases uncertainty, while entropy ejection attempts to restore order, subject to inherent system imperfections.

3.4.2. Homomorphic polynomial public key (HPPK)

HPPK was first proposed by Kuang and Perepechaenko in 2023 as a key encapsulation mechanism (KEM) [21]. In the HPPK KEM scheme, two private univariate polynomials, $f(x)$ and $h(x)$, together with two co-prime pairs, $\gcd(R_1, S_1) = 1$ and $\gcd(R_2, S_2) = 1$, form the private key. The scheme involves two entropy injection and ejection operations.

In the key generation phase, the first entropy injection operator $\hat{E}_{in}$ transforms $f(x)$ and $h(x)$ into $p(x, u_1, \ldots, u_m)$ and $q(x, u_1, \ldots, u_m)$, where $u_1, \ldots, u_m \in \mathbf{F}_p$:

$$\begin{aligned} \hat{E}_{in}|\frac{f(x)}{h(x)}\rangle &= |\frac{B(x,u_1,\ldots,u_m)f(x) \bmod p}{B(x,u_1,\ldots,u_m)h(x) \bmod p}\rangle \\ &= |\frac{p(x,u_1,\ldots,u_m)}{q(x,u_1,\ldots,u_m)}\rangle. \end{aligned} \tag{34}$$

Here, the noise polynomial $B(x, u_1, \ldots, u_m) = \sum_{i=0}^{n-1} \sum_{j=1}^{m} b_{ij} x^i u_j \in \mathbf{F}_p$ is chosen randomly. The product polynomials $p(x, u_1, \ldots, u_m)$ and $q(x, u_1, \ldots, u_m)$ can be easily derived [21]. The entropy ejection operator $\hat{E}_e$ in the decryption phase is applied as follows:

$$\hat{E}_e|\frac{c = p(x,u_1,\ldots,u_m)}{d = q(x,u_1,\ldots,u_m)} \bmod p\rangle = |\frac{f(x)}{h(x)}\rangle = |k \bmod p\rangle. \tag{35}$$

Thus, the secret $x \in \mathbf{F}_p$ can be recovered from Equation (35). Since both product polynomials remain private, they cannot be factorized back to reveal the private polynomials $f(x)$ and $h(x)$.

The second phase of entropy injection is defined as

$$\hat{p}_{in} = \frac{R_1 \cdot \square \bmod S_1}{R_2 \cdot \square \bmod S_2}, \tag{36}$$

where the bit-length L of S_1 and S_2 must satisfy $L > 2\log_2 p + \log_2 m$. The public key is then generated by applying the entropy injection operator $\hat{p}_{in}$:

$$\begin{aligned} \hat{p}_{in}|\tfrac{p(x,u_1,\ldots,u_m)}{q(x,u_1,\ldots,u_m)}\rangle &= |\tfrac{\sum_{i=0}^{n}\sum_{j=1}^{m}(R_1 p_{ij} \bmod S_1)x^i u_j}{\sum_{i=0}^{n}\sum_{j=1}^{m}(R_2 q_{ij} \bmod S_2)x^i u_j}\rangle \\ &= |\tfrac{P(x,u_1,\ldots,u_m)}{Q(x,u_1,\ldots,u_m)}\rangle. \end{aligned} \tag{37}$$

The public key consists of the following coefficients:

$$\begin{aligned} P_{ij} &= R_1 p_{ij} \bmod S_1, \\ Q_{ij} &= R_2 q_{ij} \bmod S_2. \end{aligned} \tag{38}$$

Recovering the entropy injection operators from the public key P_{ij} and Q_{ij} is computationally difficult, leading to the Hidden Modulus Product Problem (HMPP) with complexity $\mathcal{O}(2^L)$ [22].

In the encapsulation phase, given a randomly chosen secret $x \in \mathbf{F}_p$, key encapsulation is performed via entropy injection. Random noise variables $u_j \in \mathbf{F}_p$ are selected as follows:

$$\begin{aligned} \hat{p}_{encap}|P(x,u_1,\ldots,u_m)\rangle &= |\sum_{i=0}^{n}\sum_{j=1}^{m} P_{ij}[x^i \times u_j \bmod p]\rangle = |C\rangle, \\ \hat{p}_{encap}|Q(x,u_1,\ldots,u_m)\rangle &= |\sum_{i=0}^{n}\sum_{j=1}^{m} Q_{ij}[x^i \times u_j \bmod p]\rangle = |D\rangle. \end{aligned} \tag{39}$$

This results in the ciphertext pair $\{C, D\}$. The entropy injection leads to a deterministic entropy ejection with exponential complexity $\mathcal{O}(p^{m+1})$.

In the HPPK KEM, entropy ejection occurs in two stages. First, the inverse entropy injection operator $\hat{p}_{in}^{-1}$ is applied:

$$\hat{p}_{in}^{-1}|\frac{C}{D}\rangle = |\frac{R_1^{-1} \times C \bmod S_1}{R_2^{-1} \times D \bmod S_2}\rangle = |\frac{c}{d}\rangle. \tag{40}$$

with integer $C \in [0, S_1)$ and $D \in [0, S_2)$. Then, the entropy ejection operator $\hat{E}_e$ from Equation (35) is applied to remove

$$\hat{E}_e|\frac{c}{d}\rangle = |\frac{c}{d} \bmod p\rangle = |k\rangle = |\frac{f(x)}{h(x)}\rangle. \tag{41}$$

Here, the entropy injection during the encapsulation phase, $\hat{p}_{encap}$, is automatically eliminated through division in Equation (41). This allows the secret to be efficiently recovered by solving a univariate polynomial equation, especially when its order is 1. The similar injection and ejection operators can be constructed for more generic cases such as doubled-layer encryption for the public key generation to further hide the factorizable coefficients.

3.5. Entropy sources

The quality of randomness in any cryptographic system is fundamentally tied to its underlying entropy sources. Within the Quantum Cryptographic Dynamics (QCD) framework, robust entropy injection operators are constructed using entropy derived from diverse sources. This section reviews key entropy sources, including physical entropy, CPU jitter, and system jitter.

Physical Entropy Sources

Physical entropy sources exploit inherent physical phenomena like thermal noise and quantum processes. Commercial devices such as ID Quantique's Quantis [23] harness quantum mechanical effects to generate high-quality randomness with near-ideal Shannon entropy. Classical hardware random number generators (RNGs) typically rely on electronic noise; while unpredictable, this noise often requires post-processing to mitigate bias and ensure uniformity.

Quantum entropy sources leverage the intrinsic unpredictability of quantum measurements [24–28], producing outputs with extremely high entropy. Quantum Key Distribution (QKD) protocols [5, 6, 29, 30] also utilize quantum entropy for secure key establishment, though their primary role is not direct randomness generation.

CPU Jitter as an Entropy Source

CPU jitter arises from variations in execution times due to dynamic hardware interactions, such as branch prediction and memory access delays [31–33]. Gutterman et al. [34] demonstrated

that these timing variations can be exploited to extract sufficient entropy for cryptographic purposes. Precisely measuring fluctuations in CPU cycles allows systems to generate unpredictable seeds, essential for secure entropy injection.

System Jitter as an Entropy Source

System jitter encompasses a broader range of unpredictable interactions than CPU jitter, arising from operating system scheduling, memory management, and other hardware-level events. For instance, the Linux random number generator (LRNG) collects entropy from sources like CPU jitter, user inputs, disk I/O timings, and hardware interrupts [34]. These inputs are pooled together and processed using cryptographic hash functions. While the LRNG is widely used, its entropy estimation is heuristic, and it lacks a formal model describing entropy accumulation dynamics.

More recently, Red Hat's JitterEntropy RNG [32] was certified under NIST SP800-90B and incorporated into the Linux kernel as a system-level entropy source. It relies exclusively on CPU timing variations and includes formal statistical validation. NIST-approved deterministic random bit generators (DRBGs) [35] operate based on seed entropy with deterministic expansion but require careful reseeding policies and trust in the entropy source's initial state.

QPP-RNG and Quantum Cryptographic Dynamics

Kuang and Lou recently proposed a system-level entropy harvesting method—the Quantum Permutation Pad Random Number Generator (QPP-RNG) [36]—based on repeated random permutation sorting. In this scheme, a disordered array $\{|c_i\rangle\}$ is modeled as the result of applying a permutation $\hat{p} \in S_n$ to an initially ordered array $\{|a_i\rangle\}$, such that

$$\{|c_i\rangle\} = \hat{p}\{|a_i\rangle\}.$$

The goal is to recover the ordered array by applying the inverse permutation, which is modeled through a sequence of randomly selected sorting permutations:

$$\hat{p}^{-1} = \prod_{j=1}^{n_p} \hat{p}_j.$$

This sequence, driven by system jitter—measured as the time per sorting convergence—forms the Quantum Permutation Pad (QPP). The QPP acts as both a decryption mechanism and an *entropy amplifier via iterative permutations*, expanding runtime entropy via the combinatorial uncertainty of the permutation space. Convergence is defined as the event in which the array is restored to its original ordered state, with the number of permutations n_p mapped to an 8-bit output as $n_p \bmod 256$.

Within the Quantum Cryptographic Dynamics (QCD) framework, this process embodies an entropy evolution cycle. The entropy injection operator $\hat{p}$ increases system entropy through permutation randomization:

$$|c\rangle = \hat{p}|m\rangle, \quad H(|c\rangle) > H(|m\rangle),$$

Meanwhile, the entropy ejection operator, expressed as the product $\prod_j \hat{p}_j$, performs an inversion:

$$|m\rangle = \left(\prod_{j=1}^{n_p} \hat{p}_j\right)|c\rangle.$$

This formalism aligns with QCD's second and third laws; entropy increases under external injection and is reversibly extracted through decryption operations.

QPP-RNG has been benchmarked under the NIST SP800-90B IID framework across multiple platforms (macOS, Windows, and Raspberry Pi), achieving min-entropy values consistently above 7.86 bits per byte. These results are comparable to leading quantum hardware RNGs-such as IDQ Quantis, which achieves approximately 7.87 bits per byte, and substantially outperform embedded or CPU-based sources, including the Microchip ECC608 with 4.05 bits per byte and Red Hat's jitter entropy generator with 7.45 bits per byte [37–39]. Detailed benchmarking results are provided in [40].

Compared to existing RNGs, QPP-RNG offers several distinguishing features. Unlike LRNG and JitterEntropy, which either mix multiple entropy sources or rely solely on CPU jitter, QPP-RNG explicitly links entropy accumulation to a well-defined computational process—permutation sorting—with a theoretically grounded entropy growth rate of $\log_2(n!)$ per iteration. This yields a transparent and mathematically quantifiable entropy injection model. While NIST DRBGs depend on high-entropy seeds and require periodic reseeding to maintain security, QPP-RNG supports continuous entropy refresh as an intrinsic property. Moreover, unlike quantum hardware RNGs that offer excellent entropy quality but require specialized components, QPP-RNG exploits ubiquitous system behavior to approximate quantum-level unpredictability without the need for external hardware.

Summary

In summary, by harnessing a diverse array of entropy sources—from physical noise to CPU and system jitter—the QCD framework facilitates the construction of rigorously defined entropy injection/ejection operators. QPP-RNG, as an instantiation of this framework, offers a structured approach to entropy generation and utilization. Its integration with runtime system behavior enables the construction of internally self-sustaining cryptographic systems that remain resilient against both classical and quantum adversaries.

We note that detailed empirical evaluations of QPP-RNG—including tests such as NIST SP800-90B, Dieharder, and others—are presented in a separate study currently under revision at *Scientific Reports* [40]. This manuscript focuses on the theoretical framework of QCD, while practical benchmarking of QCD-based RNG implementations is addressed in dedicated works.

3.6. Implications of QCD

The key objective of Quantum Cryptographic Dynamics (QCD) as a scientific framework for cryptography is to provide rigorous guidelines for developing quantum-secure entropy injection and ejection operators. QCD abstracts cryptographic processes into fundamental physical laws governing information systems, offering the following implications.

A Unified Scientific Framework:

QCD reinterprets cryptographic operations through physical principles, establishing foundational laws that govern both encryption

(entropy injection) and decryption (entropy ejection). This framework applies to the interpretation of all cryptographic schemes, spanning classical to quantum systems and symmetric to asymmetric protocols.

Rigorous Security Analysis:

By quantifying the dynamics of entropy manipulation, QCD offers a robust basis for evaluating and comparing the security properties of cryptographic systems under quantum attack scenarios.

Guidance for Novel Protocol Design:

The QCD formalism inspires the creation of innovative cryptographic mechanisms that leverage quantum principles, ensuring enhanced security in the emerging quantum computing era.

Controlled Reversibility:

The framework emphasizes that the existence of an effective entropy ejection operator ensures reversibility, allowing injected entropy to be undone with minimal loss—a crucial property for reliable decryption.

A Systematic Approach:

QCD provides a quantifiable, methodical foundation for understanding entropy manipulation, which is key to designing robust, quantum-secure communication protocols.

In essence, while traditional cryptography focuses on the design and implementation of secure protocols, QCD serves as a higher-level scientific paradigm that underpins these protocols with fundamental principles of entropy manipulations. This framework not only aids in the rigorous analysis of security but also guides the development of quantum-resistant cryptographic primitives.

4. Conclusions

In this paper, we introduced Quantum Cryptographic Dynamics (QCD) as a unified scientific paradigm for understanding and designing cryptographic systems. By formulating cryptographic processes in terms of entropy injection and ejection, QCD establishes three fundamental laws that parallel the principles of classical mechanics, providing a novel and intuitive approach to cryptographic analysis.

The **First Law** (Entropy Inertia) asserts that an information system's entropy remains constant in the absence of external operations, emphasizing the inherent stability of data in both plaintext and ciphertext forms. The **Second Law** (Entropy Evolution) formalizes the transformation of a system's entropy through external operators, providing a rigorous analytical framework for the dynamics of cryptographic mechanisms. The **Third Law** (Entropy Redistribution) ensures reversibility in cryptographic processes, guaranteeing that well-designed encryption schemes can effectively reconstruct the original message, even in the presence of noise.

This QCD-based perspective offers a systematic approach to analyzing classical cryptographic schemes, including the One-Time Pad, AES, and public-key protocols like RSA and Diffie–Hellman, and extends naturally to post-quantum cryptographic (PQC) paradigms such as Learning With Errors (LWE), Kyber, and Homomorphic Polynomial Public Key (HPPK). By emphasizing the role of controlled entropy manipulation, QCD highlights that security fundamentally relies on the interplay of entropy injection and ejection operators.

Furthermore, we demonstrated the application of the QCD framework to the Quantum Permutation Pad Random Number Generator (QPP-RNG), interpreting it as an entropy-driven process that leverages system jitter to produce unpredictable random numbers. These entropy sources, in turn, fuel PQC key establishment schemes, forming a self-sustained quantum-secure eco-cryptosystem.

By bridging fundamental physical principles with cryptographic design, QCD provides a powerful analytical tool for security evaluation and a guiding framework for the development of quantum-resistant cryptographic primitives. As quantum computing advances, the QCD framework is poised to play an increasingly critical role in shaping secure communication protocols capable of withstanding both classical and quantum adversaries.

In summary, Quantum Cryptographic Dynamics establishes a comprehensive and principled foundation for cryptographic security, offering a clear roadmap for future research and the development of robust quantum-safe cryptographic systems.

Funding

The authors have no funding to declare.

Conflict of interest

The authors declare no conflicts of interest and have no competing interests.

Data availability statement

All data generated or analyzed during this study is included in this published article.

Additional information

Received: 2025-03-25

Accepted: 2025-07-24

Published: 2025-07-31

Academia Quantum papers should be cited as *Academia Quantum 2025*, ISSN 3064-979X, https://doi.org/10.20935/AcadQuant7841. The journal's official abbreviation is *Acad. Quant*.

Publisher's note

Copyright

References

1. 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
2. National Institute of Standards and Technology. Advanced Encryption Standard (AES), 2023. [accessed on 2025 Mar 29]. Available from: https://csrc.nist.gov/publications/detail/fips/197/final
3. 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
4. Shor PW. Algorithms for quantum computation: discrete logarithms and factoring. Proceedings of the 35th Annual Symposium on Foundations of Computer Science; 1994 Nov 20–22; Santa Fe, NM, USA; 1994. p. 124–34. doi: 10.1109/SFCS.1994.365700
5. Bennett CH, Brassard G. Quantum cryptography: public key distribution and coin tossing. Proceedings of the IEEE International Conference on Computers, Systems and Signal Processing; 1984 Dec 10–12; Bangalore, India; 1984. p. 175–9,
6. Ekert AK. Quantum cryptography based on bell's theorem. Phys Rev Lett. 1991;67(6):661–3. doi: 10.1103/PhysRevLett.67.661
7. Shannon CE. A mathematical theory of communication. Bell Syst Tech J. 1948;27(3):379–423,623–656. doi: 10.1002/j.1538-7305.1948.tb01338.x
8. Shannon, CE. Prediction and entropy of printed English. Bell Syst Techn J. 1951;30(1):50–64. doi: 10.1002/j.1538-7305.1951.tb01366.x
9. Brown PF, Pietra VJD, Mercer RL, Pietra SAD, Lai JC. An estimate of an upper bound for the entropy of English. *Comput Linguist*. 1992;18(1):31–40. https://dl.acm.org/doi/abs/10.5555/146680.146685 ISSN 0891-2017.
10. Cook JD. The most frequently used Chinese characters and their entropy. 2019. [accessed on 2025 Mar 11]. Available from: https://www.johndcook.com/blog/2019/10/18/chinese-character-entropy/
11. Olver FWJ, Lozier DW, Boisvert RF, Clark CW. NIST handbook of mathematical functions. Cambridge: Cambridge University Press; 2010; See Chapter 5 for Stirling's Approximation.
12. 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
13. Fisher RA, Yates F. Statistical tables: for biological, agricultural and medical research. Edinburgh: Oliver and Boyd; 1938.
14. Paar C, Pelzl J. Understanding cryptography: a textbook for students and practitioners. Berlin/Heidelberg: Springer; 2010.
15. Katz J, Lindell Y. Introduction to modern cryptography. 3rd ed. Boca Raton (FL): CRC Press; 2020.
16. Diffie W, Hellman M. New directions in cryptography. IEEE Trans Inf Theory. 1976;22(6):644–54. doi: 10.1109/TIT.1976.1055638
17. Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Commun ACM. 1978;21(2):120. ISSN 0001-0782. doi: 10.1145/359340.359342
18. Bennett CH, Brassard G. Quantum cryptography: Public key distribution and coin tossing. Theor Comput Sci. 2014;560:7–11. ISSN 0304-3975. doi: 10.1016/j.tcs.2014.05.025
19. Banerjee A, Peikert C, Rosen A. Pseudorandom functions and lattices. In: Pointcheval D, Johansson T, editors. Advances in cryptology—eurocrypt 2012. Berlin/Heidelberg: Springer; 2012. p. 719–37. ISBN 978-3-642-29011-4.
20. Avanzi R, Bos J, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, et al. CRYSTALS-KYBER. Specification document (update from August 2021), 2020. [accessed on 2025 Mar 30]. Available from: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf
21. 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
22. Kuang R. Quantum permutation pad for quantum secure symmetric and asymmetric cryptography. Acad Quantum. 2025;2(1):1–21. doi: 10.20935/AcadQuant7457
23. Quantique ID. Quantis quantum random number generator—technical specifications, 2018. Shannon entropy is approximately 7.999 bits per byte. [accessed on 2025 Mar 30]. Available from: https://www.idquantique.com/quantis/
24. Ma X, Yuan X, Cao Z, Qi B, Zhang Z. Quantum random number generation. NPJ Quantum Inf. 2016;2(1):16021. doi: 10.1038/npjqi.2016.21

25. Mannalatha V, Mishra S, Pathak A. A comprehensive review of quantum random number generators: concepts, classification and the origin of randomness. Quantum Inf Process. 2023;22(12):439. ISSN 1573-1332. doi: 10.1007/s11128-023-04175-y

26. Haider Z, Saeed MH, Zaheer ME, Alvi ZA, Ilyas M, Nasreen T, et al. Quantum random number generator (qrng): theoretical and experimental investigations. Eur Phys J Plus. 2023;138(9):797. ISSN 2190-5444. doi: 10.1140/epjp/s13360-023-04421-3

27. Moeini M, Akbari M, Mirsadeghi M, Naeij HR, Haghkish N, Hayeri A, et al. Quantum random number generator based on LED. J Appl Phys. 2024;135(8):084402 . ISSN 1089-7550. doi: 10.1063/5.0188208

28. Cheng J, Liang S, Qin J, Li J, Yan Z, Jia X, et al. Semi-device-independent quantum random number generator with a broadband squeezed state of light. *NPJ Quantum Inf*. 2024;10(1):20. doi: 10.1038/s41534-024-00814-z

29. Zapatero V, van Leent T, Arnon-Friedman R, Liu W, Zhang Q, Weinfurter H, et al. Advances in device-independent quantum key distribution. *NPJ Quantum Inf*. 2023;9:10. doi: 10.1038/s41534-023-00684-x

30. Yang J, Jiang Z, Benthin F, Hanel J, Fandrich T, Joos R, et al. High-rate intercity quantum key distribution with a semiconductor single-photon source. *Light Sci Appl*. 2024;13:150. doi: 10.1038/s41377-024-01488-0

31. Agafin S, Krasnopevtsev A. Memory access time as entropy source for rng. Proceedings of the 7th International Conference on Security of Information and Networks, SIN'14; 2014 Sep 9–11; New York, NY, USA; 2014. p. 176–9, Association for Computing Machinery. ISBN 9781450330336. doi: 10.1145/2659651.2659695.

32. Müller S. Cpu jitter based non-physical true random number generator. Technical report, chronox, 2014. [accessed on 2025 Feb 6]. Available from: https://www.chronox.de/jent/CPU-Jitter-NPTRNG-v2.2.0.pdf

33. Mankier. jitterentropy(3) manual page, n.d. 2025. [accessed on 2025 Feb 14]. Available from: https://www.mankier.com/3/jitterentropy

34. Gutterman Z, Pinkas B, Reinman T. Analysis of the Linux random number generator. Proceedings of the 2006 IEEE Symposium on Security and Privacy (S & P'06); 2006 May 21–24; Washington, DC, USA; 2006. p. 15–385. doi: 10.1109/SP.2006.5

35. Barker E, Kelsey J. NIST special publication 800-90A rev. 1: recommendation for random number generation using deterministic random bit generators. vol. 6. Technical report SP 800-90A rev. 1.Gaithersburg (MD): National Institute of Standards and Technology (NIST); June 2015. doi: 10.6028/NIST.SP.800-90Ar1

36. Kuang R, Lou D. Iid-based qpp-rng: A random number generator utilizing random permutation sorting driven by system jitter. arXiv. 2025. https://arxiv.org/abs/2502.18609.

37. NIST Cryptographic Module Validation Program. Entropy source validation report for ID quantique—certificate E.63. Technical report. National Institute of Standards and Technology. 2020. [accessed on 2025 May 8]. Available from: https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/entropy/E63_PublicUse.pdf

38. NIST cryptographic module validation program. Entropy source validation report for microchip ECC608 NRBG—Certificate E.46. Technical report, National Institute of Standards and Technology, 2023. [accessed on 2025 May 8]. Available from: https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/entropy/E46_PublicUse.pdf

39. NIST cryptographic module validation program. Entropy source validation report for quintessencelabs Q5 RTX—Certificate E.54. Technical report, National Institute of Standards and Technology, 2024. [accessed on 2025 May 8]. Available from: https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/entropy/E54_PublicUse.pdf

40. Vrana G, Lou D, Kuang R. Raw qpp-rng randomness via system jitter across platforms: a nist sp 800-90b evaluation. Sci Rep. 2025;15:27718. doi:10.1038/s41598-025-13135-8