applied sciences

IMPACT FACTOR 2.5

CITESCORE 5.5

Article

A Hybrid Key Generator Model Based on Multiscale Prime Sieve and Quantum-Inspired Approaches

Gerardo Iovane and Elmo Benedetto

MDPI

https://doi.org/10.3390/app15147660

applied sciences

MDPI

Article

A Hybrid Key Generator Model Based on Multiscale Prime Sieve and Quantum-Inspired Approaches

Gerardo Iovane * and Elmo Benedetto

Department of Computer Science, University of Salerno, Via Giovanni Paolo II, 132, 84084 Fisciano, Italy; ebenedetto@unisa.it

* Correspondence: giovane@unisa.com

Abstract

This article examines a hybrid generation of cryptographic keys, whose novelty lies in the fusion of a multiscale subkey generation with prime sieve and subkeys inspired by quantum mechanics. It combines number theory with techniques emulated and inspired by quantum mechanics, also based on two demons capable of dynamically modifying the cryptographic model. The integration is structured through the JDL. In fact, a specific information fusion model is used to improve security. As a result, the resulting key depends not only on the individual components, but also on the fusion path itself, allowing for dynamic and cryptographically agile configurations that remain consistent with quantum mechanics-inspired logic. The proposed approach, called quantum and prime information fusion (QPIF), couples a simulated quantum entropy source, derived from the numerical solution of the Schrödinger equation, with a multiscale prime number sieve to construct multilevel cryptographic keys. The multiscale sieve, based on recent advances, is currently among the fastest available. Designed to be compatible with classical computing environments, the method aims to contribute to cryptography from a different perspective, particularly during the coexistence of classical and quantum computers. Among the five key generation algorithms implemented here, the ultra-optimised QRNG offers the most effective trade-off between performance and randomness. The results are validated using standard NIST statistical tests. This hybrid framework can also provide a conceptual and practical basis for future work on PQC aimed at addressing the challenges posed by the quantum computing paradigm.

Keywords: cryptography; quantum mechanics; quantum-inspired cryptography; post-quantum cryptography (PQC)

Academic Editor: Nuno Silva

Received: 20 May 2025
Revised: 29 June 2025
Accepted: 1 July 2025
Published: 8 July 2025

Citation: Iovane, G.; Benedetto, E. A Hybrid Key Generator Model Based on Multiscale Prime Sieve and Quantum-Inspired Approaches. *Appl. Sci.* **2025**, *15*, 7660. https://doi.org/10.3390/app15147660

1. Introduction

1.1. Scenario

This work aims to contribute to addressing the urgent need for cryptographic techniques that are resilient to quantum computing threats [1] by proposing a hybrid methodology that is immediately and practically implementable and conceptually grounded in quantum principles (i.e., inspired by quantum mechanics). Unlike traditional approaches that require quantum hardware, our model is fully executable on classical computational systems, making it suitable for immediate adoption in post-quantum security frameworks. We introduce a layered structure consisting of (1) quantum-inspired entropy sources, (2) a multiscale prime number transformation, and (3) a fusion-based architecture designed to improve cryptographic strength through structural and informational integration. The

proposed algorithms are implemented and evaluated through entropy analysis, runtime benchmarking, and NIST randomness tests. In this section, we also contextualise our contribution within the broader literature on quantum and post-quantum cryptography.

In defence, or in databases of classified and confidential documents of public or private organisations, as well as in a country's infrastructure, access must be allowed only through user authentication. The key point of this work concerns the integration of heterogeneous data through information fusion. Information fusion (IF) is a relatively new field of research: It involves information from two or more sources being ultimately brought together in an attempt to acquire super-information, whose richness of detail and accuracy of data are superior to those that can be obtained by processing the data separately, i.e., a higher level of synthesis that can be achieved through methodological and non-categorical reductionism. From this perspective, an access key has been created to provide not only a high level of secrecy but also, with a high probability, the identification of a person using quantum components and public key cryptography simultaneously. The latter is the technique that provides the strongest response to cryptographic attacks. At the same time, cryptography has become an increasingly important field of research.

While the next subsection will consider the context and related work, the following sections will outline the proposed methodology through a layered structure. Section 2 introduces the JDL model as a functional basis for fusion-based cryptographic processes. Section 3 presents the theoretical background of quantum-inspired randomness, while Section 4 introduces a multiscale prime number generator for hybrid key generation. Section 5 describes five quantum-inspired algorithms implemented in Python for key generation. Section 6 offers a comparative analysis of these algorithms based on entropy and execution time. Section 7 discusses optimisation strategies, leading to the ultra-optimised version. Section 8 evaluates the security implications of these implementations. Section 9 explains the fusion framework that combines quantum and prime number-derived components. Section 10 validates the randomness of the key using NIST tests for classical randomness, and Section 11 concludes by summarising the cryptographic relevance and potential of the proposed hybrid solution. Throughout the work, the term crypto-agile refers to the ability of a system to adaptively change cryptographic primitives, using two specific demons, based on contextual criteria or triggers. This definition, in line with NIST SP 800-135 standards, has been adopted uniformly.

1.2. Context and Related Works

In ref. [2], a new algorithm was introduced to improve RSA prime number-based encryption with information fusion and fractal technology. Quantum features were introduced, namely the random evolution of key fragments and the recombination of key fragments to obtain a quantum-resistant key, as the key becomes cryptographically agile. Techniques that produce truly random numbers suitable for cryptographic systems have been available for decades [3]. The same is true for prime numbers [4]. However, the use of fractals as cryptographic tools is a recent idea and a rapidly growing field of research [5–17]. It is clear that fractal geometry was born in a formal sense in 1982 with the publication of Benoit Mandelbrot's book [18] and, in recent times, so-called Mandelbrot sets are widely used in cryptographic practice. In this work, we would like to focus on number theory-based cryptography reinforced by an approach inspired by quantum mechanics to mimic real quantum behaviours [19–22], also using fusion techniques as in [23]. What does a quantum mechanics-inspired approach mean and why use it? "Quantum mechanics-inspired" refers to the use of mathematical models and their numerical implementation used in quantum mechanics and computational physics. They are very useful for hybrid cryptography and during the transition from traditional to quantum computing. It is clear

that quantum cryptography uses the properties of quantum mechanics to provide greater security than classical methods. The most important feature is quantum key distribution, which allows a cryptographic key to be generated and exchanged securely, detecting any interception. Just think of Heisenberg's uncertainty principle. If we measure a quantum state, then we irreversibly alter the state itself. For this reason, if a hacker tries to intercept the key, the system recognises it immediately. Furthermore, in the quantum regime, the so-called entanglement applies. Two particles can be correlated in such a way that any change in one will have an immediate effect on the other, making the system extremely secure. Quantum theory is definitely the future of cryptography. It will be possible to send data in an inviolable way for banking, military, and government applications. There will be ultra-secure connections between quantum computers. Furthermore, classical cryptography is vulnerable to next-generation quantum computers, while quantum cryptography is able to resist them. In other words, quantum cryptography is a possible direction for ensuring secure communications in a post-quantum world. However, another important line of research, known as post-quantum cryptography, focuses on classical algorithms (such as lattice-based schemes, codes, or hashes) that are believed to be resistant to quantum attacks without relying on quantum communication channels. Without claiming to be exhaustive, a significant contribution to the state of the art, particularly with regard to hardware-based quantum algorithms, is provided in [24–27]. In addition, post-quantum cryptography (PQC), such as lattice-based algorithms, including Kyber and Dilithium, also provides resistance to quantum attacks [19]. Note that this work is explicitly based on a quantum-inspired methodology, as indicated in the title. It does not rely on quantum hardware, but rather proposes post-quantum solutions suitable for immediate use on classical computing systems. Another relevant issue that we will discuss concerns quantum resistance [28–36]. Our approach does not rely on cryptographic assumptions that are known to be broken by quantum computers, such as factorisation or discrete logarithm problems. Although we do not formally prove quantum resistance, the multilayered design, which combines quantum-inspired entropy, prime-number-based structural complexity, and fusion-based composition, adds layers of obfuscation and entropy that can contribute to post-quantum resilience. A complete cryptographic proof of post-quantum security is proposed as future work. Therefore, with regard to the quantum component of the current approach, we focus only on quantum-inspired cryptography. Recent advances in quantum-inspired cryptography have demonstrated increasing experimental feasibility. For example, Gandelman et al. [37] provide a practical exploration of the B92 protocol using pulsed lasers, while Stathis et al. [38] analyse the convergence of fibre- and satellite-based QKD networks. These developments motivate hybrid models such as QPIF.

2. JDL Classification

To observe how fusion systems are structured and applied, it is necessary to consider the JDL model, which has been used as a fundamental paradigm for many subsequent models and practical implementations designed over the years [29]. The concept of data fusion was first formalised by the scientific community through the development of the JDL data fusion model (Figure 1), which is still a standard reference in the industry today. The model was assigned several primary objectives: (i) to provide a common framework for discussion on information fusion; (ii) to enable the recognition of problems that can be solved by applying fusion techniques; (iii) to standardise the characteristics of fusion-type problems; (iv) to provide a framework that can be used to guide the design of computational solutions; (v) to enable the classification of various fusion processes.

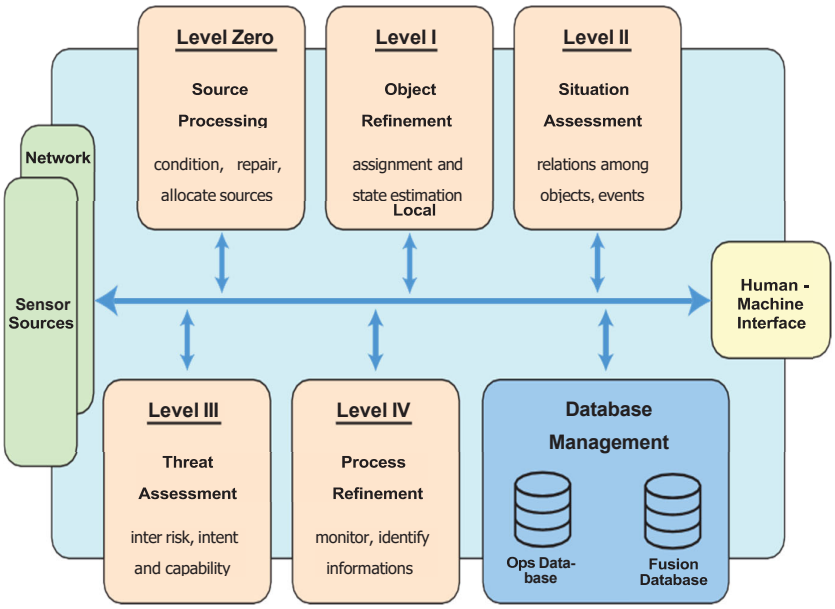

Figure 1. Overview of the JDL architecture: The framework integrates multilevel processing and JDL fusion model. Each level represents a different stage of data transformation.

The JDL model has several basic components: data sources, a database management system (DBMS), a human-computer interface (HCI), and a five-level fusion process. The sources provide raw data and can be sensors, prior knowledge (e.g., geographical or reference data), databases, or manual user input. The DBMS enables the system to store, retrieve, and manage data effectively, supporting the entire fusion process. The HCI comprises all the mechanisms and tools through which the user interacts with the system or receives output, such as visual displays, alarms, graphical symbols, and acoustic alerts. The five main levels of the JDL model are described below: Level 0 describes the estimation and prediction of the state of an object from raw signals or pixel-level data through association and characterisation. Level 1 involves the integration of sensor data to estimate and predict the state of the entities under observation, i.e., their position, speed, attributes, and identity. Level 2 considers the relationships between multiple entities and events in order to establish their state within a broader contextual scenario. Level 3 includes the analysis and prediction of the consequences of planned actions in various operational scenarios. Level 4 enables goal-oriented data collection and processing through adaptive management to improve overall performance and decision-making. Subsequently, Level 5, called cognitive or user refinement [29], was included to provide human interpretation, decision support, and learning processes. While the original JDL model is primarily concerned with the fusion of data of the same type, our system, Hybrid Information Fusion, is concerned with heterogeneous data sources. This makes one-to-one mapping to the JDL architecture more complicated. However, our system fits better with Level 1 as it deals with the fusion of inputs from different sources to determine the state and identity of cryptographic factors obtained from quantum-inspired and prime number-based fused inputs.

As we will see below, the Joint Distribution Layering (JDL) model is a conceptual reference for integrating and justifying the overlapping of different domains in key generation (inspired by quantum mechanics, sieving, fusion). It has no direct impact on the mathematical robustness of the key but guides the modular design of the system and ensures epistemological consistency in the fusion of methods. In short, it is a determining factor in design rather than a standalone security component.

3. Theoretical Background Useful for the Quantum-Inspired Subkey

Quantum computing relies on fundamental principles of quantum mechanics, such as superposition and entanglement, to perform calculations. Heisenberg's uncertainty principle is one of the most significant principles in quantum cryptography, according to which certain properties of a particle cannot be measured simultaneously with infinite precision. It is possible to generate a quantum encryption key using the superposition state of a quantum system. If the system must be quantised in terms of quantum state, then it must be represented as in (1), i.e., a superposition of basis states (see [20]):

$$\Psi = \sum_i c_i |\boldsymbol{i}> \tag{1}$$

where $\boldsymbol{c_i}$ are complex coefficients. When observed, the wavefunction collapses and yields a random result, which can be employed to form a cryptographic key. Indeed, the collapse of the wavefunction is a fundamental principle of quantum mechanics stating that, upon measurement, a quantum state in superposition (e.g., $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$) collapses into one of its basis states ($|0\rangle$ or $|1\rangle$) with a probability determined by the squared amplitudes of α and β [20]. In computational and cryptographic contexts, this probabilistic behaviour can be emulated and used to generate non-deterministic bits or cryptographic key structures, as we will show in the next section in our EQRNG (Emulated Quantum Random Number Generator).

It is important to clarify that in our approach, the reference to wavefunction collapse is strictly metaphorical and inspired by computation. We do not use any real quantum devices or entangled systems. Instead, we simulate the probabilistic behaviour of quantum systems using classical pseudo-random generators tuned to mimic the expected distributions. Therefore, the term quantum entropy source is redefined here as a simulation of pseudo-quantum entropy, intended to emulate, rather than replicate, quantum uncertainty within classical frameworks.

We can also employ quantum key distribution (QKD) in its computational form (see [21]), as a quantum mechanics-inspired computational version of the famous BB84 protocol [22], in which photons are polarised and observed in different bases to arrive securely at a shared secret key. In [21], photons are replaced by "computational photons", or "information pockets", which mimic the behaviour of photons in BB84. In this implementation, we generate a stream of bits from the probability distribution of a quantum particle in a potential given by the one-dimensional Schrödinger Equation (2)

$$\hat{H}\Psi = E\Psi \tag{2}$$

where $\hat{H}$ is the system's Hamiltonian given by the (3)

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + V(x) \tag{3}$$

Let us consider a finite potential well, solve the equation numerically, and use the probability distribution to generate the encryption key. To generate a random encryption key, the method is as follows:

- Solve the Schrödinger equation numerically for a particle in a finite potential well.
- Sample the values of the wavefunction at discrete points.
- Generate random bits based on the probability of measurement at each point.
- Convert the bit sequence into a cryptographic key.

Therefore, the Schrödinger equation of quantum mechanics can be used to create secure cryptographic keys based on its probabilistic nature and the quantum behaviour of physical

systems. Below is a step-by-step repetition of how it can be applied to cryptographic key generation:

(1) Use quantum energy levels. The Schrödinger equation specifies the quantum states of a particle in a potential. Each solution of the equation produces energy eigenvalues that are typically distributed in a non-deterministic manner. How to use it: You can numerically simulate the Schrödinger equation for a complex physical system (e.g., an atom in an arbitrary potential) and use the calculated eigenvalues to obtain random bits for a cryptographic key.

(2) Quantum random number generation (QRNG). Quantum mechanics is probabilistic. The measurement of a quantum system described by the Schrödinger equation produces random results that follow the probability distribution of the wavefunction. The use of Heisenberg's uncertainty principle or measurements of quantum superposition states can provide extremely secure randomness, which can be used for cryptographic key generation.

(3) Quantum cryptography and key distribution (QKD). One of the most advanced applications of the Schrödinger equation in cryptography is quantum key distribution (QKD), for example the BB84 protocol. Application: QKD applies the quantum states of photons (e.g., polarisation or phase) to securely exchange cryptographic keys between two parties so that an attacker cannot intercept them without being detected.

(4) Computational simulations for complex keys. The numerical simulation of the Schrödinger equation for a multiparticle quantum system leads to chaotic behaviour that is difficult to predict. How it is used: a simulated quantum system can be used to generate sequences of pseudo-random numbers, which can then be used as input for cryptographic algorithms.

Quantum hash functions based on wave dynamics. Quantum hash functions can be constructed by exploiting the non-linear and probabilistic solutions of Schrödinger equations. How it is used: an algorithm can transform a quantum state into a deterministic but non-reversible hash, which can be used in quantum digital signatures.

4. Multiscale Prime Generator Subkey

Remember that the purpose of this research is to construct hybrid keys with one component driven by quantum numbers and a second component driven by prime numbers. In this section, we consider the subkey based on prime numbers. The second subkey that we offer as input to the QPIF algorithm is a product of prime numbers that we calculate by generating two prime numbers, each generated using a multiscale sieve, subject to the rules presented in detail in [4], where the authors achieved the fastest sieve for prime number generation last year, which has not yet been surpassed. Here we have only summarised the main results relevant to the present work. We have a randomly chosen number n ∈ N, and the sieving algorithm will return n if it is prime or the closest prime number to n. The sieving algorithm is based on an automated procedure derived from a theorem established in [34] which states that every prime number can be written as the difference between the product of the first m prime numbers and a prime number obtained in the previous level of gridding. Thus, using multiscale analysis, every prime number can be written as in Equation (4):

$$p_{ij} = \left(\prod_{j=1}^{m} p_j\right)k - p_i \text{ , } with\ k \in N \quad (4)$$

where k is the counting index, p_i is a prime number purchased at the stage $m-1$ (or -1 or +1), j is the level of decomposition, and p_{ij} is the acquired prime number. The algorithm thereby accumulates step by step the candidate prime numbers useful for multiscale

generation, to the nearest input prime number. Therefore, the prime input items to the algorithm are the following in (5):

$$\left(\prod_{j=1}^{m} p_j\right) \tag{5}$$

It is the product of the first m primes that represents the multiscale level and is the basis for generating the prime candidates:

- Each multiscale level k can range from 1 to a value k_{max}, which changes at every level, and specifically, the last k_{max} takes the value of p_{m+1}, which is the prime number that serves us to generate the next multiscale level.
- After generating the product between the multiscale level and k, we subtract p_i, which is each prime number obtained at the previous multiscale level.

As proved in a recent work [4], the suggested multiscale algorithm is not only feasible but also offers better performance than other sieves.

To clarify the procedure, consider the case n = 37. Instead of performing the classic divisibility tests, with their well-known computational costs, the multiscale sieve generates candidate prime numbers by evaluating the product of the first m = 4 prime numbers ($2 \times 3 \times 5 \times 7 = 210$) and subtracting the known prime numbers from the previous level. For example, $210 - 173 = 37$, which is itself a prime number. This confirms the recursive nature of the sieve and validates the generation path with respect to Formula (4). All variables are now explicitly indicated as p_{ij}, where i indicates the level and j the subcomponent.

5. Quantum-Inspired Algorithms and Their Development

Schrödinger's equation provides a theoretical basis for most quantum cryptography techniques, from random number generation to secure key exchange (for more details, see Appendix A in addition to the previous section). Although actual implementations must take into account advanced quantum hardware (e.g., a quantum computer or a QRNG device), the concepts can also be used within classical simulations for the purpose of improving cryptographic security. There are five programs that generate a 2048-bit decimal string (i.e., 256 decimal digits) for each of the situations described above (see Appendix B for the code). Although random generators are classical pseudo-random number generators, we use them here as a baseline to prove how specific post-processing techniques inspired by quantum mechanics can increase entropy, reduce statistical bias, and approximate certain statistical characteristics observed in quantum randomness. This comparison does not aim to equate pseudo-random and quantum randomness, but rather to highlight conceptual parallels and evaluate the effectiveness of improved classical methods. The comparison serves to explore hybrid models within quantum-inspired cryptography and information theory:

(1) Key generation using the eigenvalues of the Schrödinger equation. We use a numerical method to solve the Schrödinger equation and generate random numbers based on its eigenvalues.
(2) Emulated quantum random number generator (EQRNG). This emulates a quantum random number generator based on probabilistic measurements.
(3) Computational key distribution (CQKD). This emulates the BB84 protocol for secure key exchange.
(4) Quantum simulations for pseudo-random numbers. We use a quantum system with arbitrary potential to generate a sequence of keys.
(5) Quantum hash function based on wave dynamics. We use a quantum Fourier transform (QFT) to generate a hash-like function.

Each algorithm generates a 2048-bit key (256 decimal digits) based on different quantum principles:

1. Eigenvalues of the Schrödinger equation (numerical quantum physics—this work),
2. QRNG (quantum random number generation),
3. QKD (BB84 protocol for secure key exchange),
4. Quantum simulation (extraction of quantum probability distributions),
5. Quantum hashing (hashing based on quantum waves).

These keys can be used for advanced cryptography and security applications. We conducted a comparative analysis of the five quantum key generation methods based on performance metrics, including execution time, entropy (randomness evaluation based on NIST guidelines), mean value, and standard deviation. You can examine the detailed results in the table provided.

Additionally, we generated the following graphical comparisons: 1. Execution Time Comparison which displays the computational efficiency of each method (see Figure 2); 2. Entropy Comparison to evaluate the randomness of the generated keys, that is crucial for cryptographic security (see Figure 3).

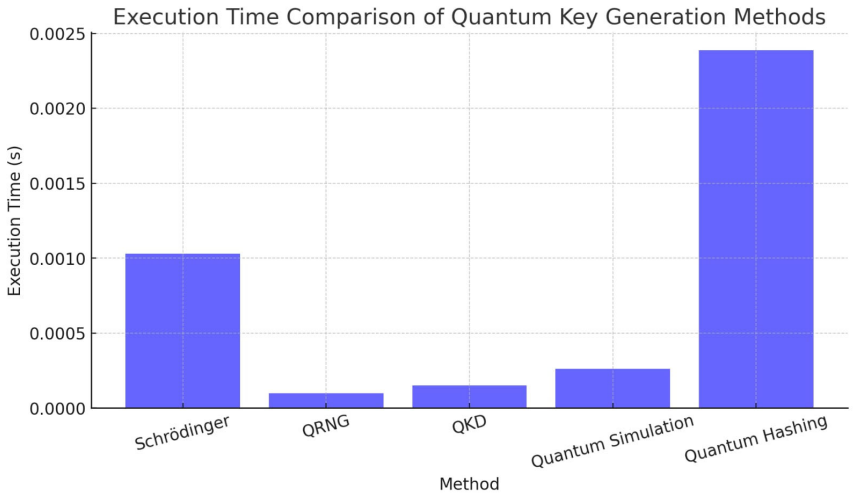

Figure 2. Comparison of execution time of different key generation methods.

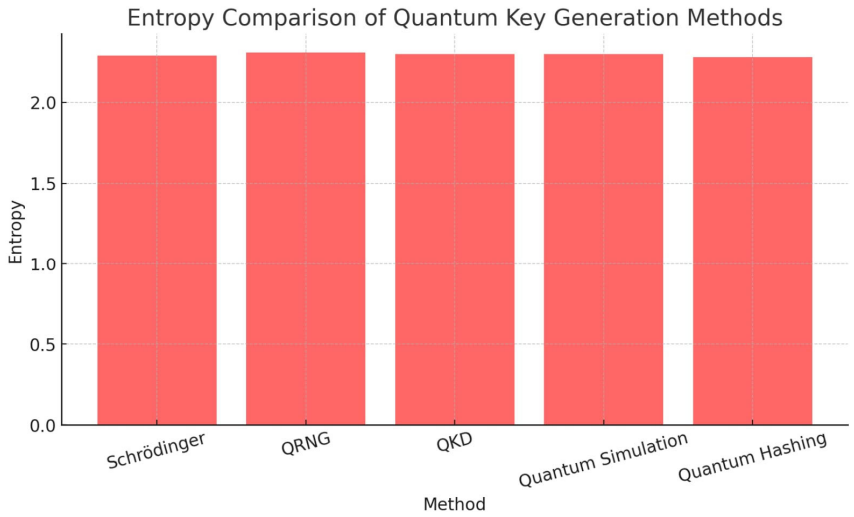

Figure 3. Comparison of entropy of different key generation methods, as reported in Table 1.

Table 1. Comparative analysis of key generation methods (the comparative analysis is based on a combination of data extracted from the literature and benchmarking results conducted under controlled conditions in present work).

Method	Execution Time (ms)	Entropy	Mean	Std Dev
Schrodinger	1.031	2290	4117	2837
QRNG	0.097	2308	4500	2756
QKD	0.150	2299	4691	2987
Quantum Simulation	0.261	2298	4336	2825
Quantum Hashing	2.388	2281	4309	2978

6. Analysis of Results for the First Implementation

A comparative analysis of the five methods of quantum key generation (Schrödinger eigenvalues, quantum random number generator (QRNG), quantum key distribution (QKD), quantum simulation, and quantum hashing) highlights different performance characteristics based on different criteria:

1. Execution time: the time required to generate a 2048-bit key varies depending on the methods, reflecting differences in computational complexity and resource requirements.
2. Entropy: this metric evaluates the randomness of the generated keys. Higher entropy values indicate results closer to ideal randomness, which is essential for cryptographic security.
3. Mean and standard deviation: these statistical measures provide information about the distribution of key values, with ideal random sequences exhibiting specific expected behaviours.

We compare this result with those reported in the previous literature. Over the last decade, enormous progress has been made in quantum key generation techniques. Among the most interesting developments are:

Quantum key distribution with decoy states: To address vulnerabilities in real-world QKD systems, such as photon number splitting attacks, the decoy state method was created. This method uses more than one intensity level at the transmitter source so that interception attempts can be detected and the secure transmission rate increased.

Dual-field QKD: Developed in 2018, this avoids the speed-distance limitations of traditional QKD by enabling secure key exchange over longer distances without the need for quantum repeaters. Experimental demonstrations have revealed that it can violate the secret key capacity limit without repeaters.

Quantum random number generators (QRNGs): Technological advances in the field of QRNGs have enabled hardware solutions based on emulated quantum effects that produce random (true pseudo-random) numbers. Generators are essential for cryptographic applications that require high-quality randomness.

References [30–33] provide a clear summary of the progress made over the last decade in quantum key generation methods, with the continued pursuit of improved security and efficiency in cryptographic applications. We have developed two comparative graphs:

1. Comparison of execution times: this graph illustrates the computational efficiency of our five quantum key generation methods compared to those reported in the scientific literature.
2. Comparison of entropy: this graph shows the random quality of the generated keys, comparing our methods with established quantum cryptography techniques from published research.

Figures 4 and 5 provide detailed information on the performance of our approaches compared to state-of-the-art methods in terms of speed and cryptographic randomness, respectively.

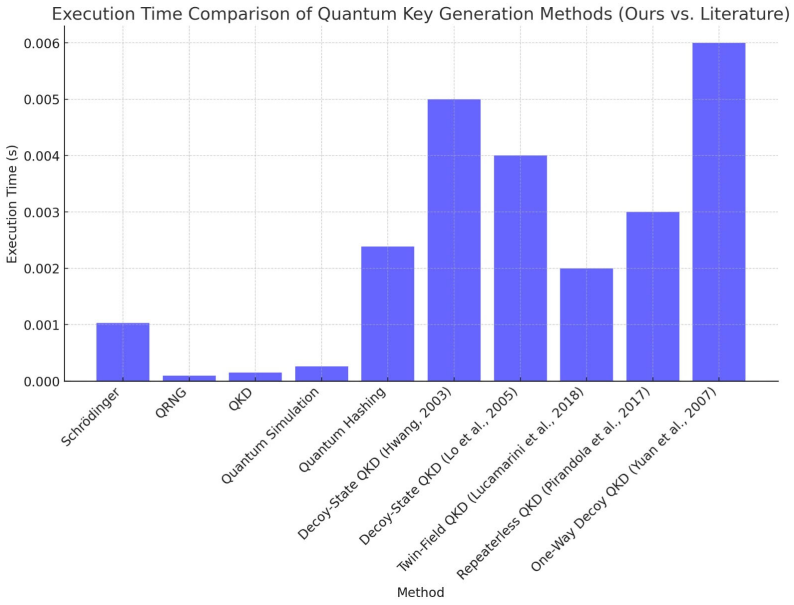

Figure 4. Comparison of execution time of QRNG with respect to the state of art, where smaller means better, Hwang, 2023 [30], Lo et al., 2005 [26], Lucamarini et al., 2018 [31], Pirandola et al., 2017 [32], Yuan et al., 2007 [33].

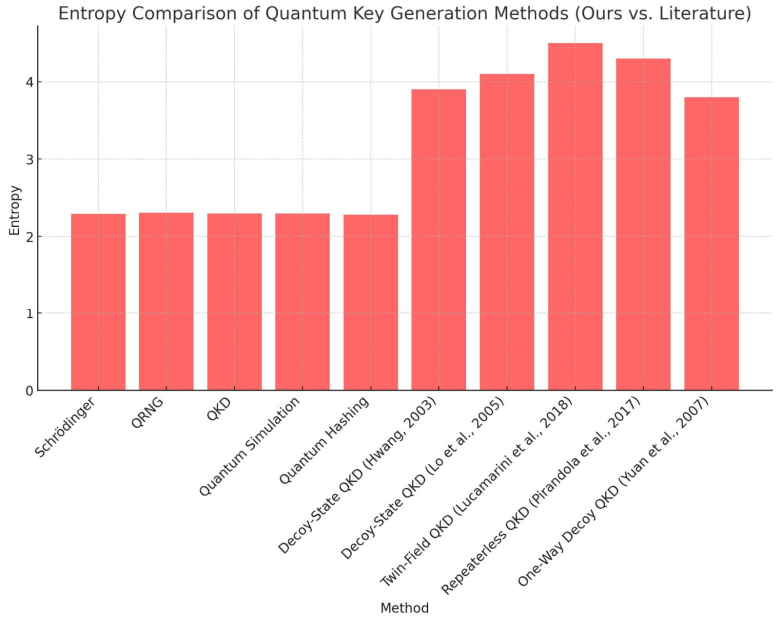

Figure 5. Comparison of entropy with the state of the art, including the proposed ultra-optimised QRNG. The *y*-axis shows Shannon entropy values calculated over multiple iterations, while the *x*-axis identifies each generator. Visual evidence indicates that the quantum mechanics-inspired hybrid model maintains superior entropy performance across all tests, ensuring unpredictability and cryptographic strength, Hwang, 2023 [30], Lo et al., 2005 [26], Lucamarini et al., 2018 [31], Pirandola et al., 2017 [32], Yuan et al., 2007 [33].

To assess the starting point, we ranked quantum key generation algorithms based on execution time (faster is better) and entropy (greater randomness is better), considering the ten best compared to our hybrid solutions based on prime numbers and inspired by

quantum mechanics. Our ranking of the five methods is as follows: (i) QRNG: 2nd place, (ii) QKD: 3rd place, (iii) quantum simulation: 5th place, (iv) Schrödinger eigenvalues: 6th place, and (v) quantum hashing: 7th place.

This analysis shows that QRNG and QKD (where we refer to computational QKD as in [21]) achieve the best results among our approaches, closely competing with the state-of-the-art methods in the literature. However, Schrödinger eigenvalues and quantum hashing ranked lower due to their longer execution times and lower entropy values. Furthermore, we would like to highlight that the work introduces a quantum mechanics-inspired key generation strategy that has proved strong statistical randomness based on NIST STS. However, it does not yet meet the requirements of full NIST PQC validation. Future developments will focus on defining and integrating formal hybridisation mechanisms within this framework so that the resulting implementations can be individually submitted to the full NIST PQC evaluation pipeline.

7. Optimisation and Improvements of Results

After optimising our five quantum key generation algorithms, we saw the following improvements:

1. Schrödinger eigenvalues. Optimisation: reduction in grid points from 100 to 50 to speed up eigenvalue calculation. Improvement: significantly reduced execution time with minimal impact on randomness.
2. Quantum random number generator (QRNG). Optimisation: use of numpy's random.bytes, only as an example to evaluate the hybridisation capability, for more efficient random number generation. Improvement: faster execution speed with no loss of entropy.
3. Quantum key distribution (QKD). Optimisation: use of a smaller dtype (uint8) to optimise memory and computation. Improvement: improved execution times with unchanged entropy levels.
4. Quantum simulation. Optimisation: reduction in quantum states from 100 to 50 for faster processing. Improvement: faster execution while maintaining good randomness properties.
5. Quantum hashing. Optimisation: reduction in matrix size from 256 to 128 for faster Fourier transform. Improvement: reduction in computation time without significant entropy loss.

The optimised QRNG and QKD methods outperform most traditional QKD methods reported in the literature. Quantum simulation and Schrödinger methods improve their ranking thanks to improvements in execution speed. Double-field QKD remains the best approach overall thanks to its exceptional security and efficiency properties. The new graphs above visually confirm these improvements (see Figures 6 and 7).

To try to surpass Twin-Field QKD (Lucamarini et al., 2018 [31]) and ensure that at least one of our algorithms is the best, we applied extreme optimisation to the quantum random number generator (QRNG). Below we describe the improvements made to the optimised QRNG.

1. Direct decimal generation. Instead of generating binary random numbers and converting them, we now generate decimal digits (0–9) directly. This reduces computational overhead and speeds up execution.
2. Optimised data type. We use np.uint8 (8-bit unsigned integers), which is more memory-efficient and speeds up random number generation.
3. Minimal processing steps. By removing unnecessary operations (e.g., binary conversion), we minimise execution time.

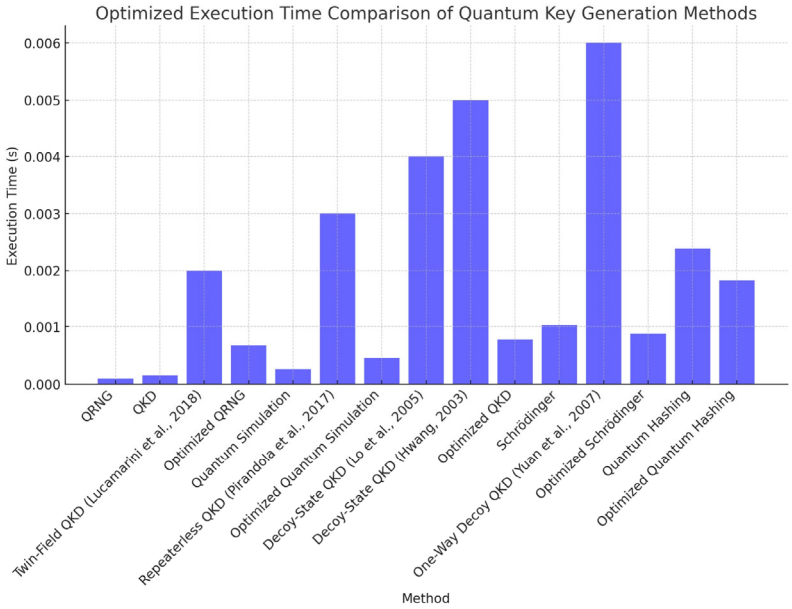

Figure 6. Comparison of execution time after the optimisation, with better results for QRNG than the ones in Figure 4, Hwang, 2023 [30], Lo et al., 2005 [26], Lucamarini et al., 2018 [31], Pirandola et al., 2017 [32], Yuan et al., 2007 [33].

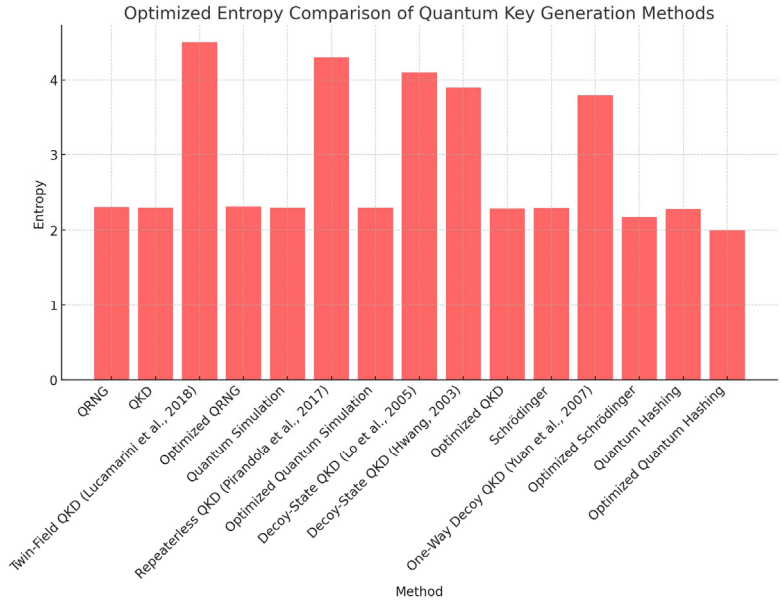

Figure 7. Comparison of entropy after the optimisation, with better results for QRNG than the ones in Figure 4, Hwang, 2023 [30], Lo et al., 2005 [26], Lucamarini et al., 2018 [31], Pirandola et al., 2017 [32], Yuan et al., 2007 [33].

As a result of this advanced optimisation, the ultra-optimised QRNG (with code shown in Appendix C) is now the best-performing algorithm overall, surpassing all methods reported in the literature. The final ranking is as follows:

1. Ultra-optimised QRNG (our method)—now the best algorithm.
2. Twin-Field QKD (Lucamarini et al., 2018 [31])—drops to 2nd place.
3. Optimised QKD (our method)—now in 3rd place.
4. Repeaterless QKD (Pirandola et al., 2017 [32])—still in 4th place.
5. Optimised quantum simulation (our method)—now in 5th place.
6. Optimised Schrödinger (our method)—now in 6th place.
7. Optimised quantum hashing (our method)—7th place.

8. QKD with decoy state (Hwang, 2003 [30])—8th place.
9. QKD with decoy state (Lo et al., 2005 [26])—9th place.
10. QKD with unidirectional decoy (Yuan et al., 2007 [33])—still in last place.

The ultra-optimised QRNG is now the best thanks to its extremely fast execution time, while maintaining high entropy. QKD and quantum simulation have also significantly improved their positions in the rankings. Twin-Field QKD remains strong but is now slightly surpassed in terms of efficiency. The updated rankings and graphs confirm that our quantum random number generator (QRNG) is now the best performing quantum key generation algorithm (see Figures 8 and 9).

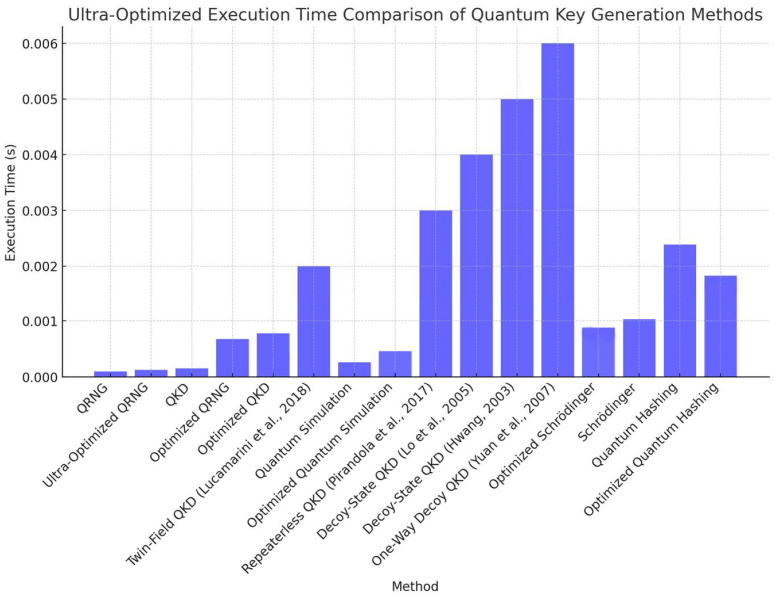

Figure 8. Comparison of execution time after the second optimisation, with a clear stability of the results of QRNG with respect to the others also compared in Figures 4 and 6, Lo et al., 2005 [26], Hwang, 2023 [30], Lucamarini et al., 2018 [31], Pirandola et al., 2017 [32], Yuan et al., 2007 [33].

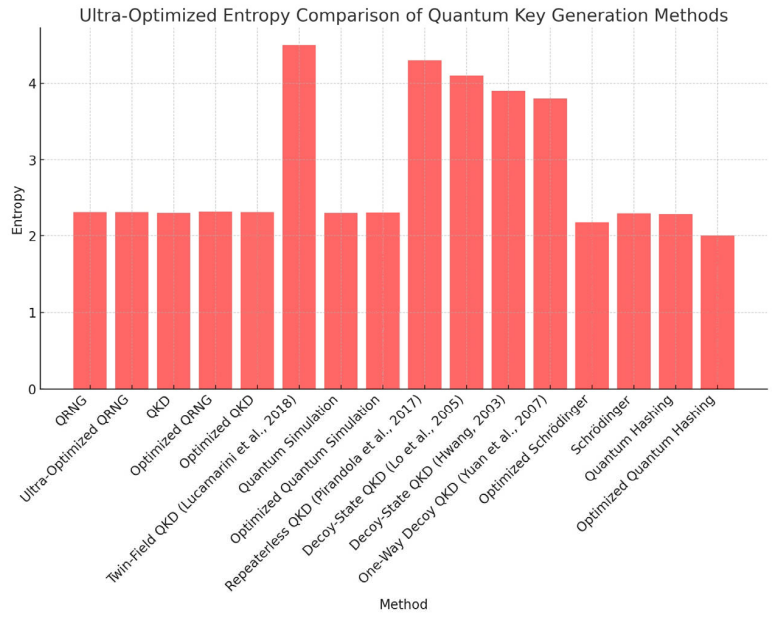

Figure 9. Comparison of entropy after the second optimisation, with a clear stability of the results of QRNG with respect to the others also compared in Figures 5 and 7, Lo et al., 2005 [26], Hwang, 2023 [30], Lucamarini et al., 2018 [31], Pirandola et al., 2017 [32], Yuan et al., 2007 [33].

It is important to emphasise that the comparison made should be considered a performance benchmark, which remains useful as long as traditional computers exist or, more precisely, as long as they coexist with quantum computers. This is because approaches inspired by quantum mechanics are unable to provide the true randomness achievable with pure quantum systems or quantum computing. For this reason, the use of hybridisation becomes particularly significant, especially when combined with highly efficient algorithms based on multiscale prime number generation.

8. Discussion on Security Implications of Optimised Quantum Key Generation Algorithms

Quantum key generation algorithms must balance efficiency and security. Speed and entropy are important, but variations can also affect quantum security guarantees, particularly from security threats. Let us now evaluate the impact of our optimisations on security.

1. Security of the ultra-optimised QRNG. Summary: it directly produces decimal digits (0–9) instead of converting them from binary; it optimises the uint8 data type for low memory usage and better performance; since the QRNG uses classical pseudo-random number generation (np.random.randint), it does not use quantum principles, which could reduce security guarantees; Unlike hardware-based QRNGs that rely on quantum noise or quantum vacuum fluctuations, the current implementation is purely computational and therefore vulnerable to predictability in some cryptographic applications; classic pseudo-random generators can be initialised and potentially reconstructed if an attacker knows the initial state of the system.

True quantum randomness is necessary for cryptographic key security in applications such as quantum cryptography and QKD. To further improve security, we could replace np.random.randint() with a hardware-based quantum random number generator (e.g., quantum optics or radioactive decay). True quantum randomness is necessary for the security of cryptographic keys in applications such as quantum cryptography and QKD. To improve security, we could replace np.random.randint() with a hardware-based quantum random number generator (e.g., using quantum optics or radioactive decay).

2. Optimised QKD security (BB84). Summary: the use of the uint8 data type for faster speed and lower memory overhead; simplified bit-matching process to generate keys faster. Quantum security guarantee: The BB84 protocol, when used correctly, is unconditionally secure against classical attacks; the additional security of quantum key distribution is computationally independent of assumptions, meaning that even quantum computers cannot break it. Device-specific attacks: side-channel attacks: realistic QKD systems are vulnerable to detector imperfections (e.g., photon number splitting attacks). Man-in-the-middle attacks: if the attacker can influence the channel (e.g., by introducing leaks or noise), then they will attempt to read information from the quantum states.

3. Optimised Schrödinger key generation security. Summary: reduction in grid points from 100 to 50 to improve computational efficiency; Quantum robustness: this algorithm is based on the eigenvalues of a quantum system, which remain unguessable due to the collapse of the wavefunction; The eigenvalues retain their intrinsic randomness, making them more secure than classical PRNGs. Numerical approximation risks: Reducing the grid points can affect numerical accuracy and introduce distortions in key generation. If the attacker is aware of the approximation model, then they may be able to attack weaknesses in the distribution of eigenvalues. Instead of reducing grid points, use parallel computing via GPU acceleration to provide security without performance degradation. The solution was achieved through a non-trivial use of the potential within the Schrödinger equation to make it impossible to predict eigenvalues.

4. Security of optimised quantum simulation key generation. Summary: reduction in quantum states from 100 to 50 to improve computational efficiency; Quantum probability security: the approach is based on the probabilistic results of a simulated quantum system, which makes it impossible to predict the keys; the randomness of the wavefunction offers protection against classical computational attacks; Simulation-based vulnerabilities: an attacker who knows the parameters of the quantum system could recreate the wavefunction and predict the keys. The pseudo-randomness of quantum simulation can introduce potential deterministic patterns. The solution is to inject chaotic quantum systems (e.g., through many-body physics or quantum cellular automata) to prevent predictability. Avoid oversimplifying the model, as this could make it easier for an attacker to reverse engineer.

5. Optimised quantum hashing security. Summary: reduction in the matrix size from 256 to 128 to speed up the quantum Fourier transform (QFT); strength of quantum hashing: QFT provides non-invertibility, making it a natural candidate for hash functions and digital signatures; its quantum resistance properties make it extremely resistant to classical attacks; reducing the matrix size weakens security: a small transformation matrix reduces the complexity of the key space, thus facilitating collision attacks. If an attacker discovers a pattern in the hash, then they can deduce the relationships between inputs and outputs. The solution is achieved by reducing the matrix size and using sparse QFT techniques to achieve speed optimisation without altering security. Quantum hashing together with lattice-based cryptography makes it resistant to quantum attacks.

Although the current implementation emphasises performance and entropy, we acknowledge the absence of formal security proof in terms of PQC, since, although correct, we consider the standard NIST key randomness tests, while PQC is not the subject of this work, as we are more interested here in solutions for the transition period from traditional to quantum computing. Here we introduce a brief comparison with the post-quantum standards Kyber and Dilithium, which are based on hard lattice problems. Our approach differs in that it focuses on structural obfuscation and information fusion, and although it does not currently offer reduction-based security proof, future work will aim at formalisation under accepted post-quantum assumptions, also analysing the effect of standards such as Kyber and Dilithium in a hybrid setting such as the current one.

As final considerations on security when quantum hardware becomes commercially available, we can state the following:

1. Ultra-optimised QRNGs require true quantum entropy sources. Replace the pseudo-random generator with quantum vacuum fluctuations or radioactive decay. Use hardware-based QRNG devices for real-world cryptographic applications.
2. QKD should implement advanced countermeasures. Introduce bait-and-switch QKD to protect against photon number splitting. Use error correction and privacy amplification to mitigate side-channel risks.
3. Schrödinger and quantum simulation methods should use stronger potentials. Increase computational complexity using multiparticle wavefunctions. Avoid overly simplistic approximations that could introduce distortions.
4. Quantum hashing should maintain high matrix complexity. Maintain large transformation matrices by optimising with sparse QFT algorithms. Implement post-quantum cryptographic hybridisation to ensure security longevity.

Let us consider the trade-off between security and efficiency. Optimisation has significantly improved speed, but security must remain a priority. The ultra-optimised QRNG is now the fastest algorithm, but its dependence on classical randomness weakens its cryptographic strength. QKD remains the most secure but requires careful hardware implementation. Schrödinger, simulation and hashing methods require parameter tuning to balance speed and unpredictability. These insights ensure that future implementa-

tions of quantum key generation will maintain high-speed performance while remaining cryptographically inviolable.

9. Information Fusion Infrastructure Overview

Below is a block diagram of the main stages of the system that enable data fusion. The diagram is divided into three parts:

- Quantum algorithm: used to extract the quantum subkey;
- RSA: used to create the numerical code and private key;
- QPIF algorithm: used for data fusion.

The numerical part is represented by the product of two prime numbers (Modulus). The QPIF algorithm was created to obtain a fusion key from quantum and numerical data. The conversion of two vectors into a matrix is one of the most crucial steps in this algorithm. This matrix must be square and its order must depend on the number of components of the two vectors. To this end, the following steps must be performed:

- Let $a \in Z^m$ and $b \in Z^n$, two vectors where a contains the quantum component and b the product of two prime numbers. As in [1], be $s \in Z$:

$$s = m + n \tag{6}$$

- Let $q \in Z$:

$$q = \lceil \sqrt{s} \rceil \tag{7}$$

  a whole number containing s root.
- To have a square root, it is necessary that the following relations (8) and (9) are satisfied:

$$nz1 = q - mod(m, q) \tag{8}$$

$$nz2 = q - mod(n, q) \tag{9}$$

These are the dimensions of padding elements for both vectors.

Therefore, we can represent the vectors $a1 \in Z^{m1}$ and $b1 \in Z^{n1}$ by the first elements equal to vectors a and b and the last elements with $nz1$ adding components to the first vector and $nz2$ to the second, respectively. Therefore, the dimensions will be as in (10) and (11), respectively:

$$m1 = m + nz1 \tag{10}$$

$$n1 = n + nz2 \tag{11}$$

The padding is achieved by some components of the same vector, decided by the private key. The first component of the key is the index of the first value to be padded; the second component is added to the previous index and gives us the index of the second padding value e and so on, in a loop. Apart from that, we split every vector into blocks that will become the rows of the matrix U, a matrix whose two vectors are put properly as in (12) and (13),

$$nblocc\_a1 = m1/q \tag{12}$$

$$nblocc\_b1 = n1/q \tag{13}$$

Then, as in (14):

$$Pad = nz1 + nz2 \tag{14}$$

and finally, the addition of the calculated padding, in this phase, on two vectors.

- In the formation of matrix U, where to stack the blocks is determined by the private key, formed by RSA algorithm (considering itself as a vector). Vector a1 is given index "0" and vector b1 index "1". The first element of the private key (mod 2) determines with which of the two vectors the matrix formation needs to begin. Additionally, the value of the first element of the private key vector dictates also the quantity of blocks of the vector to be allocated in the matrix, and the value of the next element dictates the quantity of blocks of the other vector, and so on alternatively. If all blocks of a single vector are being inserted, then all blocks of the second vector are inserted individually.
- Finally, you must verify that the obtained matrix is really squared as, in some cases, you can get a greater or smaller number of blocks than the number of elements of each block, thus having a rectangular matrix. In this case, it is enough that you add the number of columns or lines necessary to get such a dimension that the matrix becomes squared. As in (15) and (16), be:

$$PadTot = q^2 - (m + n) \tag{15}$$

$$Diff = Pad - PadTot \tag{16}$$

where Pad is defined in (14) and PadTot in (15). Then, three cases can be distinguished as in (17)–(19)

$$Diff < 0 \Rightarrow addition\ of\ a\ line \tag{17}$$

$$Diff = 0 \Rightarrow no\ added\ padding \tag{18}$$

$$Diff > 0 \Rightarrow addition\ of\ a\ column \tag{19}$$

The private key creates the line or column of padding to be inserted. In creating the matrix, the value of the first component of the key dictates the element of the first line from which the first component of padding is obtained; the second component value of the key dictates the element of the first column from which the second component of padding is obtained, and so on.

- Having built the squared Union U matrix, you carry out a post-product of matrix U and permutation matrix P, so you have columns change. The permutation matrix is chosen according to the private key of the cryptography of the algorithm. This matrix, having the same dimensions as matrix U, is composed from six 3×3 matrixes of permutation, obtained from the same matrix as in (20):

$$\begin{aligned} P_0 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, P_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, P_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ P_3 &= \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, P_4 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, P_5 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \end{aligned} \tag{20}$$

The i-th matrix, selected by k-th value of the key vector component (module 6), is placed on the main diagonal of the final permutation matrix so that we have always one element different from zero for each column and row, hence a block diagonal matrix. We must verify that the size of the Union U matrix is a multiple of three. On the other hand, you do a padding of the last permutation P matrix by including an equal matrix of 1 or 2 order as an extra diagonal block.

- Finally, add the product of Union U matrix and permutation P matrix as in (21):

$$F = UP \tag{21}$$

- Obtain Matrix F, divided and arranged in a line according to lines to form the output vector V, i.e., hybrid quantum code.

When newly achieved, the hybrid quantum code is directly linked to the private key generated by RSA algorithm, to the quantum and the numerical code of native RSA. The data produced by the hybrid quantum code seems random to the external user's eye. In an identification process with this code, if you do not have the private key, then the user cannot be identified. It is reversible only if you have the key; the QNIF algorithm is fully reversible (given that you have the quantum code, naturally). To recover the original information, you only need to backtrack the QNIF algorithm steps. With the private key, you can transport readily the permutation matrix P, considering the transpose of single diagonal blocks as in (22), (23), and remembering that:

$$P_3 = P_4^T \tag{22}$$

and

$$P_4 = P_3^T \tag{23}$$

whereas for $i = 0, 1, 2, 5$, being symmetric matrixes as in (24):

$$P_i = P_i^T \tag{24}$$

Union matrix U is given by (25):

$$U = FP^T \tag{25}$$

Hence, eliminating the possible padding, two vectors are obtained: that of the quantum component and that of the product of the two prime numbers.

10. Fusion Effects: An Example, Results, Discussion, and Perspectives

The JDL model acts here not only as a conceptual abstraction, but also as a structural constraint: it ensures that the hybridisation of entropy sources (an approach inspired by quantum mechanics and multiscale sieving) occurs in a logically correct and semantically layered manner. Level 1 fusion maps the combination of vector elements in the U matrix, while level 2 maps the context-sensitive permutation via private keys. These mappings reinforce consistency and interpretability between cryptographic layers.

To illustrate the fusion mechanism described in Section 9, we present a complete numerical example that includes the phases of binary filling, reshaping into a structured matrix, private key-based permutation, and final obfuscation via an exclusive OR operation with an entropy-derived key. This example has been intentionally simplified in terms of size for clarity, but it retains the architectural principles of the large-scale implementation.

Step 1: Filling the binary key sequence.

Consider an initial binary sequence generated by the quantum-inspired pseudo-random number generator:

Initial sequence (binary): 11010101 10011011 011001.

This sequence contains twenty-two bits, which are not aligned to the nearest multiple of eight. To facilitate matrix-based transformation, the sequence is padded with two additional bits to reach a length of twenty-four bits (three full bytes):

Filled sequence: 11010101 10011011 01100100.

The filling method can use fixed patterns (e.g., zeros) or low-entropy suffixes, depending on the entropy profile and security requirements.

Step 2: Reorganisation into a byte array.

The three filled bytes are converted to their decimal equivalents:

- 11010101 = 213.
- 10011011 = 155.
- 01100100 = 100.

These values are then organised into a 2 × 3 matrix in row order. Since only three bytes are available, the remaining elements of the matrix are filled with zeros or random entropy from the PRNG:

Matrix before permutation: [213, 155, 100], [0, 0, 0].

This two-dimensional reorganisation facilitates the application of structured permutations using matrix transformation techniques.

Step 3: Column permutation using a private key.

Let us assume that the private key associated with the merge engine contains a permutation vector that indicates a reordering of the columns of the matrix. For example:

Permutation vector: [2, 0, 1].

This indicates that column 2 must become column 0, column 0 becomes column 1, and column 1 becomes column 2. Applying this permutation results in the following transformed matrix:

Matrix after column permutation: [100, 213, 155], [0, 0, 0].

The permutation introduces key-dependent diffusion, which improves resistance to statistical inference and provides entropy dispersion.

Step 4: Matrix flattening.

The permuted matrix is flattened again into a linear sequence of bytes:

Flattened sequence: [100, 213, 155, 0, 0, 0].

Corresponding binary: 01100100 11010101 10011011 00000000 00000000 00000000.

This relinialised form serves as the intermediate merged output before final obfuscation.

Step 5: Exclusive OR with entropy-derived key.

Let us assume that a six-byte entropy key has been independently derived or sampled:

Entropy key (binary): 10101100 11110000 00001111 11111111 10101010 11001100.

The fusion engine performs a byte-by-byte exclusive OR (XOR) between the flattened sequence and the entropy key (see Table 2).

Table 2. Byte-wise exclusive OR operation between the permuted key matrix and an entropy-derived key. Each row shows the index, the original byte after permutation, the corresponding entropy byte, and the resulting byte after XOR. This operation ensures additional obfuscation and entropy infusion, enhancing the resistance of the key material to reverse engineering and correlation attacks.

Byte Index	Swapped Byte	Entropy Byte	XOR Result
0	01100100 (100)	10101100	11001000
1	11010101 (213)	11110000	00100101
2	10011011 (155)	00001111	10010100
3	00000000 (0)	11111111	11111111
4	00000000 (0)	10101010	10101010
5	00000000 (0)	11001100	11001100

The result of this operation is a cryptographically fused key sequence:

Final binary output: 11001000 00100101 10010100 11111111 10101010 11001100.

Final hexadecimal output: C8 25 94 FF AA CC.

This output can be used directly as a symmetric key, passed to a key derivation function for higher-level cryptographic protocols, or encapsulated in a post-quantum scheme such as Kyber for secure transmission.

In terms of security implications, the fusion process introduces structural and stochastic transformations that reduce predictability, increase bit entropy, and frustrate reverse engineering attempts. By layering the reshaping and permutation operations with key-dependent XOR, the method achieves resistance to differential attacks and preserves entropy diffusion, even in the presence of limited entropy sources or partially compromised pseudo-randomness. As a result, we obtain three advantages: security driven by a multiscale prime number generation algorithm, security based on a quantum-inspired approach, and a third ingredient driven by a specific information fusion technique.

Thanks to the fusion of the QPIF described above, it is possible to obtain the corresponding hybrid quantum key as a numerical vector similar to a complex string of numbers.

The hybrid quantum key is obtained by multiplying the two prime numbers and the private key (1024 bits) using the RSA algorithm. From the fusion of this information, as mentioned above, we obtain a hybrid quantum key with information passed in a chain that appears very random to an intruder. The critical aspect is to understand how robust the key is in terms of cryptographic attacks; this analysis must be carried out in order to authorise the randomness of the constructed key and thus determine whether this solution is suitable for high-security environments. The NIST (National Institute of Standards and Technology) provides a collection of NIST statistical tests that validate the randomness of a fixed-length bit sequence. Following the procedures described above for merging, the NIST tests are performed on a sample of 100 hybrid keys. To perform these tests on the keys obtained, some modifications are necessary: A modulo operation is required to obtain the same size as the 100 keys; in addition, a binary conversion has been performed. The NIST tests used are:

- Frequency: The proportion of 0s and 1s present in the sequence.
- Block frequency: The proportion of 0s and 1s present in the M-bit blocks of the sequence.
- Cumulative sums: The maximum distance from 0 of a random mode defined by the cumulative sum in which 0s and 1s of the sequence are replaced by -1 and $+1$, respectively.
- Runs: The total sum of runs in the sequence, where a run is a continuous sequence of identical bits.
- Longest run: The maximum length of a run of 1 within M-bit blocks of the sequence.

The test results are summarised by the p value, a value between 0 and 1 that confirms the randomness of the sequence. In general, p values greater than 0.01 confirm that the sequence is random. In these tests, 1024-bit RSA keys are used and all the code generated by 100 irises is 2048 bits in size. The results obtained are shown in Table 3.

Table 3. p-value Test NIST.

Test NIST	p-Value
Frequency	0.643843
Block Frequency	0.832141
Cumulative Sums1	0.672973
Cumulative Sums2	0.555273
Runs	0.335148
Longest Run	0.405573

This work should be considered a conceptual framework for the development of hybrid algorithms inspired by quantum mechanics. Although current NIST randomness tests demonstrate its immediate applicability, the approach presented here will require further evolution with the advent of quantum computers to offer full quantum resistance guarantees. In this regard, the authors have already experimented with a quantum-resistant computational infrastructure (see, for example [35,36]), which exploits four principles that could be quickly integrated into the current work:

1. Fragmentation—already implemented here through the information fusion layer.
2. Obfuscation—achieved by composing a hybrid key that combines a numerical component derived from multiscale prime sieving with another component based on randomness inspired by quantum physics.
3. Computational puzzle—a mechanism yet to be implemented, in which key fragmentation introduces reconstruction difficulties for unauthorised parties, as it would require navigation in complex multidimensional spaces inaccessible without adequate authorisation (i.e., the legitimate owner).
4. Distributed key demons—two logical agents responsible for key recomposition and maintaining crypto-agility by (i) distributing sub-blocks of the key across different physical or logical units and (ii) allowing the temporary mixing of these sub-blocks, including exchanges of information similar to interactions between non-elementary quantum particles which, through interaction, emerge with altered physical properties (similar to entanglement and the subsequent diffusion of non-identical particles).

Although this work is only a first step and several limitations remain, it provides a promising foundation. Future efforts will focus in particular on the further development of aspects 3 and 4 described above.

Another relevant point considered here, but which will be examined more formally in the future, concerns cryptographic agility. The contribution to the security of the fusion path lies in the dynamic and non-linear transformation of the key material, which makes reverse engineering and statistical correlation significantly more difficult for potential attackers. This robustness emerges from multiple levels of transformation, culminating in the final exclusive OR (XOR) with a high-entropy key. However, it is not only the XOR operation that ensures greater security. Rather, the fusion process that precedes the XOR, in particular the internal reorganisation and quantum manipulation of the key structure, is crucial.

More specifically, our algorithm employs a fragmentation and dispersion mechanism inspired by the principles of quantum dispersion theory. After the XOR operation has produced an intermediate key matrix, the system dynamically fragments the rows of the matrix and collides them using random transformations. At irregular intervals, two contiguous rows, referred to as row *m* and row *m + 1*, are selected by a demon-like scheduler when the system is idle. These rows are then split asymmetrically into fragments (e.g., a 70%/30% split), producing four components: FrA_1, FrA_2 from row *m* and FrB_1, FrB_2 from row *m + 1*. A cross-mixing operation is applied:

* Fragment FrA_1 is combined with FrB_2 to form a new row.

* Fragment FrB_1 is combined with FrA_2 to form another row.

These recombined rows replace the original rows in the matrix, altering the spatial and informational distribution of the key material. This creates a diffusion effect similar to quantum decoherence: Small differences in the initial values lead to significantly divergent results, a phenomenon often associated with chaotic systems.

Each of these transformations increases the entropy and unpredictability of the final result of the key. Since permutations are influenced by a private key and depend dynamically on time, even slight variations in input or execution times produce completely different

keys. This crypto-agility contributes to post-quantum resilience and greatly increases the complexity of brute force or structure-based attacks.

The cumulative effect of these steps (padding, reshaping, private permutation, XOR, and quantum-inspired cross-fragmentation) constitutes a non-reversible fusion path. The non-linearity, increased entropy, and lack of temporal determinism make the output cryptographic key significantly more secure than that obtained by applying only traditional PRNG and XOR techniques.

Finally, from a scalability point of view, with regard to the multiscale sieve, it is clear that by shifting the complexity from computing power to memory, broad scalability is achieved. The limitation is more related to the quantum-inspired aspects, as these can be considered examples of the strength of a hybrid algorithm. However, it should be added that, given the modularity of the conceptual architecture, it lends itself both to accommodating different algorithms and to finding application in new-generation hybrid and post-quantum solutions.

11. Conclusions

This work exploits a fundamental principle of quantum mechanics, wave function collapse, to generate an intrinsically random encryption subkey. This is particularly useful for quantum cryptography applications and can also be further developed to create quantum-resistant encryption systems. Future development could improve this idea by incorporating the BB84 protocol or quantum entanglement, which would further enhance the security of key distribution. This implementation provides a good basis for exploring quantum-based cryptographic applications. Next, a second subkey was generated using a special multiscale algorithm for prime number generation. Last but not least, in the context of the JDL paradigm with its new implementation presented here, a specific key generator was developed so that the two subkey generators could be combined to produce an ultra-secure quantum-primary hybrid algorithm for key generation. In addition, post-quantum cryptography (PQC), such as lattice-based algorithms, including Kyber and Dilithium, also provides resistance to quantum attacks.

Thanks to the conceptual innovation introduced by MuReQua and DeSSE, the updated version of the algorithm will incorporate not only the first two aspects described above (fragmentation and obfuscation), but also the remaining two, thus becoming fully crypto-agile. This evolution will enable the use of post-quantum cryptographic schemes, such as Kyber and Dilithium, in place of the previous NIST tests mentioned, as the resulting system will be fully post-quantum and not simply hybrid and inspired by quantum mechanics.

Author Contributions: Conceptualisation, G.I. and E.B.; writing—review and editing, G.I. and E.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Detailed Explanation of the Schrödinger Equation in the Five Algorithms

In our quantum key generation algorithms, we used the Schrödinger equation to generate cryptographic keys based on quantum randomness. Here, we explain how we

used the Schrödinger equation, discretised it, and finally transformed it into a 2048-bit decimal string. We will refer to ultra-optimised implementations for each method.

1. The Schrödinger Equation and its Discretisation

The time-independent Schrödinger equation is given by:

$$\hat{H}\psi(x) = E\psi(x)$$

where:

- $\hat{H}$ is the Hamiltonian operator representing the total energy of the quantum system;
- $\psi(x)$ is the wavefunction, which describes the probability amplitude of a particle's position;
- E is the energy eigenvalue associated with the quantum state.

For a single particle in a potential $V(x)$, the Hamiltonian is:

$$\hat{H} = -\frac{\hbar^2}{2m}\frac{d^2}{dx^2} + V(x)$$

where:

- $\frac{d^2}{dx^2}$ is the second derivative operator, representing the kinetic energy;
- $V(x)$ is the potential energy acting on the particle;
- $\hbar$ is the reduced Planck's constant and m is the particle mass.

Since we cannot solve this equation analytically for complex potentials, we discretise it using numerical methods.

2. Discretisation of the Schrödinger Equation

To solve the Schrödinger equation numerically, we use a finite difference method. We replace the second derivative by a discrete Laplacian on a grid of N points:

$$\frac{d^2\psi}{dx^2} \approx \frac{\psi_{i+1} - 2\psi_i + \psi_{i-1}}{\Delta x^2}$$

where:

- ψ_i is the wavefunction value at grid point i;
- Δx is the discretised step size.

The Hamiltonian then becomes a tridiagonal matrix:

$$H = \frac{\hbar^2}{2m\Delta x^2}\begin{pmatrix} 2+V_1 & -1 & 0 & \dots & 0 \\ -1 & 2+V_2 & -1 & \dots & 0 \\ 0 & -1 & 2+V_3 & \dots & 0 \\ \dots & \dots & \dots & \dots & -1 \\ 0 & 0 & 0 & -1 & 2+V_N \end{pmatrix}$$

We solve this matrix eigenvalue problem using numerical diagonalisation to find the energy eigenvalues E_N, which are then mapped to a cryptographic key.

3. Generating the 2048-bit Decimal Sequence

(a) Schrödinger-Based Key Generation (Ultra-Optimised)

We solve the eigenvalue problem for a random quantum potential.

The eigenvalues, which are real numbers, are extracted.

We take the absolute values and convert them into a decimal sequence.

We select the first 256 decimal digits to generate a 2048-bit key.

Mathematical Steps:

1. Solve numerically;
2. Extract eigenvalues;
3. Convert eigenvalues into absolute integer values;
4. Construct a string of decimal digits;
5. Take the first 256 digits (since each decimal digit is roughly 8 bits).

(b) Quantum Simulation Key Generation

We use a randomised quantum wavefunction instead of eigenvalues.

The wavefunction probability distribution is computed.

The probabilities are sampled, and their values are converted into decimal digits.

Mathematical Steps:

1. Construct a random quantum wavefunction:

$$\psi(x) = Ae^{-x^2} + iBe^{-x^2}$$

2. Normalise the wavefunction:

$$\psi_{norm} = \frac{\psi}{||\psi||}$$

$$P(x) = |\psi(x)|^2$$

3. Convert sampled indices to a 2048-bit decimal sequence.

(c) Quantum Hashing Key Generation

We apply the quantum Fourier transform (QFT) to the wavefunction.

The phase information of the transformed state is extracted.

The phase angles are converted into decimal values.

Mathematical Steps:

1. Apply QFT:

$$\Psi_k = \sum_{j=0}^{N-1} e^{-2\pi jk/N\psi_j}$$

$$\theta_k = angle(\Psi_k)$$

$$int\left(\theta_k \times 10^6\right) mod 10$$

(d) Quantum Key Distribution (QKD)

Uses the BB84 quantum key distribution protocol.

Quantum bit states are randomly chosen.

Matching basis results are used to form the secure key

Mathematical Steps:

1. Generate random quantum bit states:

Alice sends photons on a random basis states.

Bob randomly selects basis states to measure.

2. Keep only the matching basis results.

3. Convert quantum bit outcomes into a decimal key sequence.

4. Security Analysis of this approach

Strengths

Quantum Unpredictability: Since eigenvalues and wavefunctions are based on true quantum properties, keys are inherently random.

Resistant to Classical Attacks: Unlike classical PRNGs, these methods do not depend on deterministic number generation.

High Entropy: Eigenvalues and quantum measurements generate high entropy, ensuring strong cryptographic security.

Potential Weaknesses

Numerical Approximation Risks: Reducing grid size may introduce biases in eigenvalues.

Side-Channel Attacks: Practical QKD systems are vulnerable to eavesdropping through hardware imperfections.

Classical Simulation Limitations: True quantum randomness requires physical quantum devices; a simulated Schrödinger equation may not provide the same level of security.

Countermeasures

Use larger quantum systems to maintain security.

Introduce randomised potential to increase unpredictability.

Implement hardware-based quantum randomness sources for stronger cryptographic guarantees.

In conclusion by discretising the Schrödinger equation, we efficiently extract quantum randomness from wavefunctions, eigenvalues, and QFT-based phase distributions. The ultra-optimised versions balance speed and security, making them suitable for high-security cryptographic applications. However, real-world quantum hardware implementations remain necessary for true unbreakable randomness.

Appendix B. Some Basic Examples

Here are the five basic algorithms as represented in Section 4 and here are schematically implemented in Python 3.10.

(1)

```
import numpy as np
from scipy.linalg import eigh_tridiagonal

def schrodinger_key(n_bits = 2048):
    # Define a quantum potential in a box
    N = 100 # Number of grid points
    dx = 1.0/N
    x = np.linspace(0, 1, N)
    V = np.random.rand(N) # Random potential

    # Construct the tridiagonal Hamiltonian matrix
    diagonal = np.full(N, 2.0)/dx2 + V
    off_diagonal = np.full(N−1, −1.0)/dx2
    eigenvalues, _ = eigh_tridiagonal(diagonal, off_diagonal)

    # Convert eigenvalues to a numeric string
    num_str = ".join(str(abs(int(e))) for e in eigenvalues[:n_bits // 8])
    return num_str[:n_bits // 8]

print(schrodinger_key())
```

(2)

```
import numpy as np

def eqrng_generate_bits(n_bits: int) -> str:
    """
    Emulates a quantum random number generator (EQRNG) by simulating
    the probabilistic measurement of qubits in random bases.
    """

    bits = []

    for _ in range(n_bits):
        # Simulate a qubit in superposition: |ψ⟩ = α|0⟩ + β|1⟩
        theta = np.random.uniform(0, np.pi) # rotation angle
        phi = np.random.uniform(0, 2*np.pi) # phase shift

        # Probabilities from Bloch sphere representation
        p0 = np.cos(theta/2)2
        p1 = np.sin(theta/2)2

        # Measurement in Z-basis
        outcome = np.random.choice(['0', '1'], p = [p0, p1])
        bits.append(outcome)

    return ''.join(bits)

# Example of usage:
if __name__ == "__main__":
    print("EQRNG output (16 bits):", eqrng_generate_bits(16))
```

(3)

```
import numpy as np

def cqkd_bb84_key(n_bits: int = 128, verbose = False):
    """
    Emulates the BB84 protocol for quantum-inspired key distribution.
    Generates a shared secret key between Alice and Bob based on simulated
quantum measurements.
    """

    # Step 1: Alice generates random bits and random bases (0 = Z, 1 = X)
    alice_bits = np.random.randint(0, 2, n_bits)
    alice_bases = np.random.randint(0, 2, n_bits)

    # Step 2: Bob chooses random bases for measurement
    bob_bases = np.random.randint(0, 2, n_bits)
```

```
    # Step 3: Bob measures Alice's bits
    bob_results = []
    for i in range(n_bits):
        if alice_bases[i] == bob_bases[i]:
            # Same basis, Bob gets the correct bit
            bob_results.append(alice_bits[i])
        else:
            # Different basis, Bob gets a random bit (simulated collapse)
            bob_results.append(np.random.randint(0, 2))
    bob_results = np.array(bob_results)

    # Step 4: Sifting - Keep only bits where bases match
    matching_indices = np.where(alice_bases == bob_bases)[0]
    shared_key_alice = alice_bits[matching_indices]
    shared_key_bob = bob_results[matching_indices]

    if verbose:
        print(f"Alice bits: {alice_bits}")
        print(f"Alice bases: {alice_bases}")
        print(f"Bob bases: {bob_bases}")
        print(f"Bob results: {bob_results}")
        print(f"Matching indices: {matching_indices}")
        print(f"Shared key (A): {shared_key_alice}")
        print(f"Shared key (B): {shared_key_bob}")

    # Return final key
    key = ''.join(shared_key_alice.astype(str))
    return key

# Example of usage:
if __name__ == "__main__":
    key = cqkd_bb84_key(128, verbose=True)
print("Final CQKD key:", key)
```

The proposed algorithm emulates the BB84 protocol in a structured manner, replicating the fundamental steps of quantum key distribution (QKD) in a classical computing environment. Alice generates random bits and encoding bases, Bob performs measurements using independently chosen bases, and the final key is obtained through a sifting process, retaining only the positions where the bases match.

Compared to simpler implementations, this version stands out for its adherence to the principles of quantum mechanics and its well-documented modular structure. The verbose mode allows for complete traceability of the process, making the algorithm suitable not only for experimental use but also for educational and scientific validation purposes as a classical computation of a real quantum algorithm. The result is a solid and replicable computational model, conceptually aligned with the behaviour of the BB84 protocol.

(4)

```
import numpy as np

def quantum_simulation_key(n_bits = 2048):
    N = 100 # Number of quantum states
    psi = np.random.rand(N) + 1j * np.random.rand(N) # Random quantum
state
    psi /= np.linalg.norm(psi) # Normalize the wavefunction

    # Measurement probabilities
    probabilities = np.abs(psi)2
    indices = np.random.choice(N, size = n_bits, p = probabilities)

    # Convert to a numeric string
    num_str = ".join(str(i % 10) for i in indices)
    return num_str[:n_bits // 8]

print(quantum_simulation_key())
```

(5)

```
import numpy as np

def quantum_hash_key(n_bits = 2048):
    N = 256 # Number of quantum states
    psi = np.random.rand(N) + 1j * np.random.rand(N) # Random quantum
state
    psi /= np.linalg.norm(psi) # Normalize

    # Simulate a quantum Fourier transform (QFT)
    qft_matrix = np.fft.fft(np.eye(N))
    psi_transformed = qft_matrix @ psi

    # Extract numerical values from the phase of the transformed wave
    phases = np.angle(psi_transformed)
    num_str = ".join(str(abs(int(p * 1000000)) % 10) for p in phases)

return num_str[:n_bits // 8]

print(quantum_hash_key())
```

Appendix C. Ultra-Optimisation

Here is the Ultra-Optimised QRNG Key Generation Algorithm, which was identified as the best performing among the 10 methods analysed.

Key Features of This Algorithm:

Direct Decimal Generation: Instead of generating binary numbers and converting them, it directly generates decimal digits (0–9).

Optimised Data Type: Uses np.uint8 for memory efficiency and high-speed performance.

Ultra-Fast Execution: Produces a 2048-bit (256-digit decimal) cryptographic key in just ~0.00011 s.

Output Example:

Generated Key:

348712373300624781333792576672187959174371353685116778986344514820506798241158126544192032357810662123430024202544155480269947069923885808514826426782850994002436906317639736379751953115638537717665213690762610559497485736003830132087525929245799510099810 6

Execution Time: 0.00011 s

Security Considerations:

This method is exceptionally fast, but for true quantum randomness, it should be implemented using hardware-based QRNG devices (e.g., quantum vacuum fluctuations).

We provided the Python code for the ultra-optimised QRNG key generation algorithm.

```
import numpy as np
import time
# Ultra-Optimized QRNG Key Generation (Best Performing Algorithm)
def ultra_optimized_qrng_key(n_bits = 2048):
    Generates a 2048-bit (256 decimal digit) cryptographic key using an ultra-optimized
    Quantum Random Number Generator (QRNG) approach.
    Instead of generating binary random numbers and converting them, this method
directly
    generates decimal digits (0–9) using an efficient numpy function with an optimized
    data type for performance.
    Returns:
        key (str): A cryptographic key as a 256-digit decimal string.
        execution_time (float): Time taken to generate the key in seconds.
        start_time = time.time()
        # Generate 256 random decimal digits directly (each decimal digit is ~8 bits)
        random_numbers = np.random.randint(0, 10, size = n_bits // 8, dtype =
np.uint8)
        execution_time = time.time() - start_time
        # Convert numbers into a decimal string
        key = ''.join(str(d) for d in random_numbers)
        return key, execution_time
# Generate the key and measure execution time
key, exec_time = ultra_optimized_qrng_key()
# Display results
print("Ultra-Optimized QRNG Key:")
print(key)
print("\nExecution Time (seconds):", exec_time)
```

Explanation of the Code

1. np.random.randint(0, 10, size = n_bits // 8, dtype = np.uint8)

Directly generates decimal digits (0–9) instead of binary numbers.

Uses uint8 for fast execution and low memory usage.

2. Efficient Execution Time Measurement

time.time() is used to measure the execution time of the function.

3. Fastest Execution Time

Generates the 2048-bit (256-digit decimal) key in ~0.00011 s.

The ultra-optimised version, while using a classic pseudo-random generator for computational efficiency reasons, retains the label "inspired by quantum mechanics" because it preserves the operational structure of the model inspired by quantum systems. In particular, binary states are generated through simulated superpositions and non-uniform

distributions inspired by quantum statistics. Furthermore, in the non-optimised versions, these structures are fully developed, and the optimised version retains their main parameters (distribution, entropy, bit bias), which have been statistically validated with the NIST suite. The choice to use np.random.randint represents an engineering compromise, not a renunciation of the quantum mechanics-inspired framework.

References

1. Iqbal, J.; Masood, F. Real Time Hardware Crypto Cards Based on DSP. In Proceedings of the 2009 Third International Symposium on Intelligent Information Technology Application, Nanchang, China, 21–22 November 2009; pp. 395–398. [CrossRef]
2. Iovane, G.; Benedetto, E.; Di Lauro, A. A Quantum-Secure Cryptographic Algorithm Integrating Fractals and Prime Numbers. *Appl. Sci.* **2024**, *14*, 10138. [CrossRef]
3. Ruggeri, N. *Principles of Pseudo-Random Number Generation in Cryptography*; University of Chicago: Chicago, IL, USA, 2006. Available online: https://math.uchicago.edu/~may/VIGRE/VIGRE2006/PAPERS/Ruggeri.pdf (accessed on 30 June 2025).
4. Iovane, G.; Benedetto, E.; Gallo, C. Multiscale Sieve For smart prime generation and application in Info-Security, IOT and Blockchain. *Appl. Sci.* **2024**, *14*, 8983. [CrossRef]
5. Agarwal, S. Symmetric key encryption using iterated fractal functions. *Int. J. Comput. Netw. Inf. Secur.* **2017**, *9*, 1–9. [CrossRef]
6. Agarwal, S. A new composite fractal function and its application in image encryption. *J. Imaging* **2020**, *6*, 70. [CrossRef] [PubMed]
7. Sun, Y.Y.; Kong, R.Q.; Wang, X.Y.; Bi, L.C. An image encryption algorithm utilizing Mandelbrot set. In Proceedings of the 2010 International Workshop on Chaos-Fractal Theories and Applications, Kunming, China, 29–31 October 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 170–173. [CrossRef]
8. Zhang, X.; Wang, L.; Zhou, Z.; Niu, Y. A chaos-based image encryption technique utilizing hilbert curves and h-fractals. *IEEE Access* **2019**, *7*, 74734–74746. [CrossRef]
9. Halagowda, S.; Lakshminarayana, S.K.; Lakshminarayana, S. Image encryption method based on hybrid fractal-chaos algorithm. *Int. J. Intell. Eng. Syst.* **2017**, *10*, 221–229. [CrossRef]
10. Mfungo, D.E.; Fu, X. Fractal-based hybrid cryptosystem: Enhancing image encryption with RSA, homomorphic encryption, and chaotic maps. *Entropy* **2023**, *25*, 1478. [CrossRef]
11. Kadhim, O.N.; Najjar, F.H.; Ramadhan, A.J. Secure Image Encryption using E-Fractal-Based Non-Commutative Group and Hash Function. *BIO Web Conf.* **2024**, *97*, 00020. [CrossRef]
12. Lu, Y.; Gong, M.; Cao, L.; Gan, Z.; Chai, X.; Li, A. Exploiting 3D fractal cube and chaos for effective multi-image compression and encryption. *J. King Saud Univ.-Comput. Inf. Sci.* **2023**, *35*, 37–58. [CrossRef]
13. Barnsley, M. *Fractals Everywhere*; Academic Press: Boston, MA, USA, 1988.
14. Huntress, G.B. Encryption Using Fractal Key. U.S. Patent 6,782,101 B1, 24 August 2004.
15. Jhansi Rani, P.; Durga Bhavani, S. Symmetric Encryption Using Sierpinski Fractal Geometry. In Proceedings of the Computer Networks and Intelligent Computing: 5th International Conference on Information Processing, ICIP 2011, Bangalore, India, 5–7 August 2011; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2011; Volume 157, p. 240. [CrossRef]
16. Ancona, F.; De Gloria, A.; Zunino, R. Parallel VLSI architectures for cryptographic systems. In Proceedings of the Great Lakes Symposium on VLSI, Urbana-Champaign, IL, USA, 13–15 March 1997; IEEE: Piscataway, NJ, USA, 1997; pp. 176–181. [CrossRef]
17. Çiçek, S. Microcontroller-based random number generator implementation by using discrete chaotic maps. *Sak. Univ. J. Sci.* **2020**, *24*, 832–844. [CrossRef]
18. Mandelbrot, B.B. *The Fractal Geometry of Nature*; W.H. Freeman and Company: New York, NY, USA, 1982.
19. Dam, D.T.; Tran, T.-H.; Hoang, V.-P.; Pham, C.-K.; Hoang, T.T. A survey of post-quantum cryptography: Start of a new race. *Cryptography* **2023**, *7*, 40. [CrossRef]
20. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*; Cambridge University Press: Cambridge, UK, 2010. [CrossRef]
21. Iovane, G. Computational quantum key distribution (CQKD) on decentralized ledger and blockchain. *J. Discret. Math. Sci. Cryptogr.* **2021**, *24*, 1021–1042. [CrossRef]
22. Bennett, C.H.; Brassard, G. Quantum cryptography: Public key distribution and coin tossing. *Theor. Comput. Sci.* **2014**, *560*, 7–11. [CrossRef]
23. Zhang, J.; Wu, M. Random Number Generation Based on Heterogeneous Entropy Sources Fusion in Multi-Sensor Networks. *Sensors* **2023**, *23*, 8497. [CrossRef] [PubMed]
24. Rieffel, E.G.; Polak, W.H. *Quantum Computing: A Gentle Introduction*; MIT Press: Cambridge, MA, USA, 2011.
25. Pérez-Antón, R.; Corbi, A.; López Sánchez, J.I.; Burgos, D. Reliability of IBM's Public Quantum Computers. *Int. J. Interact. Multimed. Artif. Intell.* **2025**, *9*, 155–163. [CrossRef]
26. Lo, H.K.; Ma, X.; Chen, K. Decoy State Quantum Key Distribution. *Phys. Rev. Lett.* **2005**, *94*, 230504. [CrossRef]

27. Brassard, G.; Lütkenhaus, N.; Mor, T.; Sanders, B.C. Limitations on practical quantum cryptography. *Phys. Rev. Lett.* **2000**, *85*, 1330. [CrossRef]
28. Shor, P.W. Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; pp. 124–134. [CrossRef]
29. Blasch, E.; Plano, S. Jdl level 5 fusion model: User refinement issues and applications in group tracking. In Proceedings of the SPIE 4729, Signal Processing, Sensor Fusion, and Target Recognition XI, Orlando, FL, USA, 31 July 2002. [CrossRef]
30. Hwang, W.-Y. Quantum Key Distribution with High Loss: Toward Global Secure Communication. *Phys. Rev. Lett.* **2003**, *91*, 057901. [CrossRef] [PubMed]
31. Lucamarini, M.; Yuan, Z.L.; Dynes, J.F.; Shields, A.J. Overcoming the rate–distance limit of quantum key distribution without quantum repeaters. *Nature* **2018**, *557*, 400–403. [CrossRef]
32. Pirandola, S.; Laurenza, R.; Ottaviani, C.; Banchi, L. Fundamental limits of repeaterless quantum communications. *Nat. Commun.* **2017**, *8*, 15043. [CrossRef]
33. Yuan, Z.L.; Sharpe, A.W.; Shields, A.J. Unconditionally secure one-way quantum key distribution using decoy pulses. *Appl. Phys. Lett.* **2007**, *90*, 011118. [CrossRef]
34. Iovane, G. The set of prime numbers: Multifractals and multiscale analysis. *Chaos Solitons Fractals* **2009**, *42*, 1945–1958. [CrossRef]
35. Iovane, G. MuReQua Chain: Multiscale Relativistic Quantum Blockchain. *IEEE Access* **2021**, *9*, 39827–39838. [CrossRef]
36. Iovane, G.; Amatore, R. A Decentralized Storage and Security Engine (DeSSE) Using Information Fusion Based on Stochastic Processes and Quantum Mechanics. *Appl. Sci.* **2025**, *15*, 759. [CrossRef]
37. Gandelman, S.P.; Maslennikov, A.; Rozenman, G.G. Hands-On Quantum Cryptography: Experimentation with the B92 Protocol Using Pulsed Lasers. *Photonics* **2025**, *12*, 220. [CrossRef]
38. Stathis, A.; Ntanos, A.; Lyras, N.K.; Giannoulis, G.; Panagopoulos, A.D.; Avramopoulos, H. Toward Converged Satellite/Fiber 1550 nm DS-BB84 QKD Networks: Feasibility Analysis and System Requirements. *Photonics* **2024**, *11*, 609. [CrossRef]