A Virtualized BB84 QKD Simulator in Python Using Qiskit Aer for Education and Prototyping

Mert Büyükdede
Istanbul Atlas University, Istanbul, Türkiye
E-mail: mertbuyukdede@gmail.com

Keywords: quantum cryptography, cybersecurity, quantum key distribution, python

Received: May 2, 2025

With the rapid advancement of quantum technologies, quantum key distribution (QKD) has become a key approach to secure communication. This study presents a software-only implementation of the BB84 QKD protocol developed in Python using IBM's Qiskit Aer simulator. The proposed framework is designed for educational and prototyping purposes rather than for hardware-level security evaluation. The simulator models the core stages of BB84: random bit/basis generation, state preparation, basis-dependent measurement, sifting, and parameter estimation via QBER, followed by conceptual post-processing. Evaluation reports (i) the expected sifted-key ratio under random basis selection, (ii) the ideal-channel QBER behavior under noise-free Aer execution, and (iii) the theoretical intercept–resend detection probability derived from standard BB84 analysis. For sequences of ≥100 bits, the implementation matches standard BB84 expectations under ideal Aer execution: an expected sifted-key ratio of ≈50%, QBER → 0 in the absence of adversarial interference, and an intercept–resend detection probability above 99.9% for m = 100 disclosed verification bits. These results demonstrate that an idealized virtual environment can reproduce the BB84 measurement–disturbance trade-off in the information-theoretic sense, without claiming physical decoherence modeling unless an explicit Aer noise model is enabled. The analysis focuses on idealized BB84 behavior, where disturbance arises solely from basis mismatch and intercept–resend measurements rather than from physical decoherence or hardware noise models. The framework provides a reproducible platform for teaching and early-stage experimentation, with future work targeting realistic noise models and advanced post-processing.

Povzetek: Študija predstavi programsko simulacijo protokola BB84 za kvantno distribucijo ključev, ki v idealnih pogojih potrjuje pričakovano varnostno obnašanje in služi kot izobraževalno ter prototipno orodje.

1 Introduction

The protection and encryption of data have always been crucial throughout history. With technological advancements, threats and countermeasures have grown proportionally. Since its emergence, quantum physics has fascinated physicists and other scientists. In particular, the application of quantum mechanics became a significant research focus following its introduction to the scientific community in the early 20th century [1].

The innovations brought by quantum physics have been increasingly applied to physical systems, and over time, it was proposed that quantum mechanics could be used for data transfer in communication technologies [2]. This idea led to the development of quantum cryptography and quantum key distribution concepts by the late 20th century, promising significant enhancements in communication security.

Efforts to improve the security of the communication sector, which is critically important today, continue at full speed. Encryption and data transfer based on quantum mechanics are among the systems considered for secure communication research [3]. This study is driven by the following research question: Can a fully virtualized BB84 simulation reproduce the statistical signatures used for disturbance and eavesdropping detection, without access to quantum hardware?

The primary goal is to develop a reproducible BB84 simulation environment using classical computing resources. Specifically, the study aims to (1) simulate the BB84 protocol in Python using Qiskit's Aer simulator, (2) implement a modular software workflow to support educational use and prototyping, and (3) validate the system's performance in terms of eavesdropper detection probability and practical applicability.

1.1 Cybersecurity and information security

Cybersecurity refers to protecting networks, systems, devices, or collections of devices from unauthorized access and activities such as stealing, altering, or deleting information. Cyberattacks, which pose a threat to cybersecurity, are often carried out using viruses, worms, or bots and are challenging to trace. Strategic planning is critical in cybersecurity; attackers' perspectives and thought processes are analyzed to predict their actions.

Information security is of great importance in military relations, politics, diplomacy, commerce, and communication systems. Due to increasing cyber threats, concerns about the security of these domains have arisen. Cryptography, which ensures information security by concealing data and securely transferring it, is widely used. Cryptographic systems rely on the principle that encrypted messages cannot be decoded without knowing the encryption key [4].

Cryptanalysis is the science of breaking cryptographic systems, requiring high computational power [5]. According to Moore's Law, the computational power of computers has doubled every two years since 1970 [6]. With the rising computational power of computers, cryptanalysis can now utilize the high-level mathematical computation required. Quantum computers, with their unparalleled computational capabilities, are expected to render existing encryption systems ineffective. Quantum cryptography is anticipated to be a countermeasure against this challenge.

1.2 Quantum physics and cryptology

The foundation of quantum cryptology is built upon quantum physics. Today's quantum computer is based on the application of quantum mechanics, which is a branch of quantum physics. In the early 20th century, scientists such as Albert Einstein, Niels Bohr, Werner Heisenberg, Erwin Schrödinger, Max Planck, Louis de Broglie, and Max Born contributed to the development of quantum mechanics to its current level [7].

In 1981, Richard Feynman first introduced the concept of a quantum computer during his lecture at the Massachusetts Institute of Technology [8]. Feynman stated that quantum mechanics could not be accurately simulated using classical computers, but that a new type of computer, which would be developed in the future, could simulate an atom [9].

In 1994, Peter Shor developed an algorithm based on quantum mechanics capable of breaking modern encryption systems [10]. The Shor algorithm, when executed on a quantum computer, can factorize a large number into its prime factors in a matter of seconds. Shor's algorithm is considered a milestone in the field of quantum cryptology. Since this milestone, numerous studies have been conducted in the field of quantum cryptology, starting from the 2000s to the present day.

A classical binary bit can represent only one of two states: 0 or 1. A classical bit has a single state. A quantum bit, or qubit, carries the possibility of being either 0 or 1. Unlike a classical bit, a qubit does not have a single value. A qubit exists in a state of superposition, meaning it holds both values simultaneously. This property of the qubit is linked to the principle of "superposition" in quantum mechanics. Superposition can be briefly defined as a particle being able to exist in different states at the same time [11].

1.3 Quantum algorithms

An algorithm is a step-by-step procedure to perform a calculation or solve a problem that can be executed on a computer. Therefore, when an algorithm is executed on a quantum computer, it is referred to as a quantum algorithm. In general, it is possible to run all classical algorithms on a quantum computer. However, the term "quantum algorithm" refers to those algorithms where at least one of the steps distinctly involves quantum principles such as superposition or entanglement [12]. Today, applications based on quantum algorithms exist. These applications run on quantum computers and provide acceleration or efficiency improvements over any classical algorithm. Although large-scale quantum computers are not yet available, the theory of quantum algorithms has been actively researched for over 20 years [12]. This section aims to provide an overview of the Shor and Grover algorithms, which enable quantum computers to solve complex problems in polynomial time.

1.3.1. Shor's algorithm

In 1994, mathematician Peter Shor discovered polynomial-time quantum algorithms for solving the integer factorization problem and the discrete logarithm problem, both of which are central to public-key cryptography [13]. Shor's algorithm is a quantum algorithm used to find the prime factors of a positive integer N. In his work, proved that factorizing large integers would fundamentally change with quantum computers. This algorithm is of great importance in cryptography because today's encryption systems rely on the assumption that it is computationally infeasible to factor large numbers within a reasonable time frame using classical computers. Therefore, all public-key cryptography systems currently in use will become insecure once sufficiently large quantum computers are available [13] The effectiveness of these algorithms in quantum computers is attributed to the phenomena of entanglement and parallelism in quantum mechanics [6]. Shor's algorithm is designed to handle very large numbers. Even with this algorithm, classical computers take a prohibitively long time (thousands of years, for example) to break RSA encryption. However, with quantum computers, these large numbers can be factorized in just a few minutes. Shor's algorithm will break RSA encryption and revolutionize the field of cybersecurity. For instance, most security systems in use today rely on the following principle: p and q are two prime numbers ($p \neq q$), and N = pq, where N is easy to calculate when it is small. For example, if N = 21, then p = 3 and q = 7. However, when N is the product of two 500-digit prime numbers, it is difficult to find N's prime factors. This problem cannot be solved using classical computers. However, using quantum computers and Shor's algorithm, these prime factors can be found quickly, enabling the solution of such problems.

1.3.2. Grover's algorithm

Discovered by Lov Grover in 1996, Grover's algorithm has impacted many cryptographic systems [14]. It forms the foundation of most positive applications in quantum computing [15]. Grover's algorithm provides a clever way to search for a specific element or elements that satisfy a particular query in an unsorted list. Classically, searching an unsorted database requires a linear search with time

complexity O(N). However, in Grover's algorithm, this search is performed in O(√N) time. This is the fastest possible quantum algorithm for searching an unsorted database. Unlike other quantum algorithms that can speed up classical counterparts, Grover's algorithm provides "only" a quadratic speedup. However, when N is large, even quadratic speedup is significant. Like all quantum computer algorithms, Grover's algorithm operates probabilistically, providing the correct result with high probability. Moreover, the probability of error can be reduced by repeating the algorithm [3].

With the advancement of quantum computers, it is expected that in the future, Shor and Grover's algorithms will reduce the security level of classical encryption methods. An example of how these algorithms affect classical encryption methods is presented in the table 1.

Table 1: Conceptual impact of quantum algorithms on selected classical cryptographic schemes

Classical Encryption Scheme	**Classical Security Basis**	**Effect of Quantum Algorithms (Conceptual)**
AES (256-bit key)	Security relies on brute-force resistance of symmetric keys	Grover's algorithm provides a quadratic speedup for brute-force key search, effectively reducing the security margin of a 256-bit AES key to that of a 128-bit key; AES itself is not broken, but larger key sizes are recommended for post-quantum robustness.
RSA (2048-bit modulus)	Security relies on the hardness of integer factorization	Shor's algorithm can factor large composite numbers in polynomial time *in principle*, meaning RSA would become insecure if large-scale fault-tolerant quantum computers become available. Modern RSA is indexed by modulus size (e.g., 2048/3072 bits), not "128-bit RSA."

With a sufficiently powerful quantum computer, it is observed that using Grover's algorithm, the security level of a 256-bit AES key would be reduced to the security level of a 128-bit AES key. Additionally, by using Shor's algorithm, it is possible to break a 128-bit RSA algorithm.

1.4 Classical encryption methods

Until the discovery of public key encryption methods in the 1970s, all encryption systems were based on a different principle, now known as the symmetric key encryption method. In symmetric key encryption systems, when the sender wants to send a message to the receiver, they possess an encryption key to encrypt the message, while the receiver must have a decryption key to decrypt the encrypted message. The encryption and decryption keys are the same. One simple and still highly effective symmetric key encryption system is the Vernam cipher [15]. The sender and receiver start with an identical n-bit secret key sequence. The sender encodes the n-bit message by adding the key to it, while the receiver decrypts the message by subtracting the decryption key from the encrypted message. An important feature of this system is that as long as the key sequences are genuinely secret, provable security is guaranteed. An eavesdropper can always introduce noise into the communication channel, but the sender and receiver can detect this noise and announce the disturbance. For any secret listening strategy used by the eavesdropper, the sender and receiver can guarantee that the eavesdropper's information about their unencoded messages can be made arbitrarily small. On the other hand, despite its widespread use, the security of public key encryption relies on unproven mathematical assumptions related to the difficulty of solving certain problems, such as factoring large numbers, with classical computers. The next section briefly discusses two of the most common symmetric and asymmetric encryption methods, AES and RSA.

1.4.1. RSA encryption method

RSA is the most commonly used asymmetric encryption algorithm. It uses two different keys to encrypt and decrypt messages. The algorithm was first proposed in 1976 and implemented in 1978 by Ronald Rivest, Adi Shamir, and Leonard Adleman from MIT [3]. The algorithm derives its name from the initials of these three individuals. This encryption system relies on mathematical complexity. It uses one-way mathematical functions. While it is easy to compute the "x" value of a given function, reversing this computation is extremely difficult. RSA is based on factorization. The sender selects two large prime numbers and multiplies them to obtain a large number, which is then used as the public key. The two large prime numbers are kept secret and become the private key. The security of RSA depends on the difficulty of factoring the public key into its prime components without knowing the private key.

1.4.2. AES encryption method

AES, a symmetric encryption method, is the most widely used algorithm among block cipher algorithms. It became officially adopted in 2002 and was developed from the Rijndael algorithm, becoming the American standard. AES uses the same key for both encryption and decryption. It handles a 128-bit input block and has key lengths of 128, 192, and 256 bits. Depending on the key length, the number of rounds performed during encryption also changes. For a 128-bit key, 10 rounds are used; for a 192-bit key, 12 rounds; and for a 256-bit key, 14 rounds. AES is known for its high performance in both software and hardware [16].

1.5 Limitations of classical encryption methods

As technology has advanced, the limitations of classical encryption methods have become evident. For example,

the problem of factoring large integers in a reasonable time frame can be overcome by applying Shor's algorithm on quantum computers. The biggest challenge in symmetric key encryption systems is the secure distribution of key bits [15]. The shortcomings of RSA and AES encryption methods are outlined in the Table 2.

Table 2: Selected practical limitations of RSA and AES in classical cryptographic systems

RSA	AES
Inefficient for encrypting large data volumes; typically used only for key exchange and digital signatures.	Security depends on the mode of operation (e.g., GCM, CTR, CBC with random IV). Insecure modes such as ECB expose plaintext patterns.
Requires a trusted mechanism (e.g., PKI) to authenticate public keys.	Requires secure key generation, distribution, and storage.
Security collapses if integer factorization becomes feasible (e.g., via Shor's algorithm on a large-scale quantum computer).	Vulnerable to quantum brute-force search via Grover's algorithm, which reduces the effective key strength from (k) bits to approximately (k/2) bits.

While the security of classical encryption systems is based on the difficulty of calculating some difficult mathematical problems, this technique is based on quantum mechanics. By taking advantage of this advantage, various key distribution protocols such as BB84, B92, SARG, Ekert protocol have been proposed to share equal key sets among users in a 100% secure manner.

1.6 Quantum key distribution protocols

Quantum cryptography, or Quantum Key Distribution (QKD), differs from traditional cryptographic systems in that its security is based more on physical laws than on mathematics [17]. The foundation of quantum key distribution is based on Heisenberg's uncertainty principle, a fundamental law of physics. According to this principle, it is impossible to simultaneously measure two properties (position and momentum) of a quantum system, and a measurement of one property destroys the information about the other. This principle suggests that any measurement of a quantum system like a photon will disturb its state, making subsequent measurements unreliable. The uncertainty of transmission channels through which data is transmitted becomes critical in quantum communication. If the data is altered, it indicates that an unauthorized party is attempting to intercept the transmission. Both the sender and receiver can detect such an intrusion and ensure the confidentiality of the channel [18].

1.6.1. BB84 Protocol

The BB84 Protocol was the first protocol used in quantum cryptography [8]. This protocol allows the sender and receiver to obtain a secure secret key through quantum communication, providing high security. In the BB84 protocol, the sender and receiver utilize both a quantum channel and another classical channel accessible to everyone. The quantum channel, typically created using fiber optic cables, sends photons one by one.

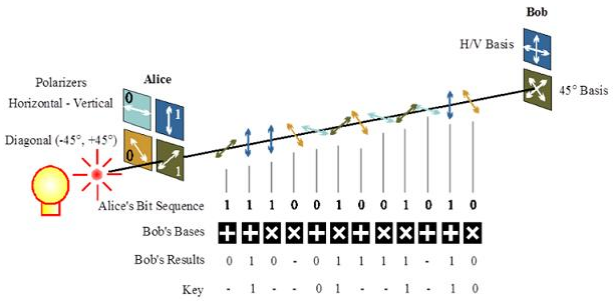

Figure 1: BB84 Quantum key distribution protocol

The BB84 protocol creates a secure key by carrying candidate bits with photon particles. The operation of the BB84 protocol is outlined below:

Step 1: The sender generates raw key bits randomly. In this simulation, randomness is produced using a classical pseudo-random number generator. The corresponding qubit states are prepared and transmitted through an idealized simulated quantum channel in Aer (noise-free unless a noise model is explicitly configured). Each bit is expressed as the polarization state of a photon, and the photons are sent one by one to the receiver over a quantum channel, which is sufficiently isolated from external interference and allows for unidirectional communication.

Step 2: The receiver measures each incoming photon in a randomly chosen basis. If the selected basis is the same as the sender's, the measurement result will match the sender's bit. If a different basis is chosen, the measurement result will have a 50% chance of being correct, but this is unknown. The same situation occurs for an unauthorized third party as well.

Step 3: After all transmission and measurements are completed, the receiver discloses, through an unsecured communication channel, which bases they used for measurement. Communication is bidirectional and authenticated, and the transmission can be passively eavesdropped. The sender then reveals the same bases they used. Ideally, the bits at these indices should be identical.

1.7 Research hypothesis and scope

This study tests the hypothesis that, under idealized simulation conditions, a software-only implementation of the BB84 protocol can reproduce the characteristic statistical signatures predicted by theory. Specifically, it is hypothesized that an intercept–resend attack yields a quantum bit error rate close to 25% in the sifted key and that the corresponding eavesdropper detection probability exceeds 99% when at least 100 verification bits are disclosed.

The scope of this work is intentionally limited to an idealized, single-node simulation environment

implemented using Python and the IBM Qiskit Aer simulator. The framework is designed for educational use and early-stage prototyping rather than for hardware-level performance evaluation or operational security claims. Physical noise sources, device imperfections, and large-scale network effects are therefore outside the scope of the present study.

2 Related works

Quantum key distribution (QKD) systems have been utilized in various regions globally to establish secure communication networks. In Tokyo, Japan, a secure communication network using quantum key distribution was implemented with the integration of six different QKD systems. Operating at GHz speeds, the quantum key distribution system enabled the first secure TV conference over a 45 km distance. In the case of detecting an eavesdropper, the system would securely redirect the communication, and key distribution would be re-established through trusted nodes [19]. In 2008, the first computer network protected by quantum key distribution was realized during a conference in Vienna. Funded by the EU, the SECOQC network connected six locations in Vienna and the town of Sankt Pölten, 69 km to the west, using standard 200 km fiber-optic cables for the connection [20]. The DARPA Quantum Network, developed between 2002 and 2007 in the United States, was the first quantum key distribution network consisting of 10 optical nodes (locations). The network, operational since late 2003, functioned continuously between 2005 and 2007, and it was compatible with the standard internet technologies of that time, allowing its use for video conferencing and other activities [21]. The SwissQuantum quantum key distribution network, established in Geneva, Switzerland, in March 2009, operated until 2011. The primary aim of the network was to test the security of quantum key distribution, and it was composed of three nodes [22]. In China, a quantum network was created in Wuhu in May 2009, featuring five nodes located in government buildings. The keys generated by the quantum key distribution network were used for encrypting video, audio, text, and confidential files exchanged between these locations [23]. The Los Alamos National Laboratory in the United States established a quantum key distribution network in 2011, composed of a central hub and associated components. Messages were routed through the central hub, which was equipped with quantum transmitters (lasers) rather than expensive photon detectors. Communication was facilitated using one-time pads (OTP), and secure communication was ensured via classical connections. The security of the entire network was reliant on the central hub being secure [24]. The QUESS space mission, launched in 2016 as a joint initiative between China and Austria, aimed to establish a communication line protected by quantum key distribution between space and Earth. The mission successfully facilitated the first intercontinental quantum video call over a 7,500 km distance [25]. The IBM Q Experience, launched in the summer of 2016, is a cloud-based quantum computing system that has been used by over 60,000 users from all seven continents for more than 1.7 million remote experiments. The system is incorporated into educational curricula and used by researchers to explore quantum computing, error correction, and validation techniques, as well as quantum measurements [26]. Table 3 below summarizes key features of previous QKD implementations and contrasts them with the current virtualized simulation approach. Unlike physical systems that rely on expensive quantum hardware and real-world transmission media, our method operates entirely in a simulated environment using IBM's Aer simulator.

Table 3: Summary of Related Quantum Key Distribution (QKD) Implementations

Study / Project	**System Type**	**Transmission Medium**	**Noise Model**	**Scale / Infrastructure**	**Purpose / Scope**
Tokyo QKD Network [19]	Hardware-based	Fiber optics (45 km)	Physical channel noise	Metropolitan network	Operational secure communication
SECOQC (Vienna) [20]	Hardware-based	200 km fiber optics	Physical noise	Multi-node network	Large-scale QKD deployment
DARPA Quantum Network [21]	Hardware-based	Optical fiber (10 nodes)	Realistic noise	Multi-node testbed	Experimental QKD infrastructure
SwissQuantum [22]	Hardware-based	Fiber optics	Controlled field noise	3-node network	Long-term performance evaluation
QUESS Satellite [25]	Space-based hardware	Satellite–Earth	High-loss channel	Intercontinental	Space-based QKD
This Study (Virtualized)	Simulated (IBM Aer)	No physical medium	None (idealized)	Single-node simulation	Education and prototyping

3 Methodology

This study implements an idealized BB84 workflow using Python and IBM's Qiskit Aer simulator. The description covers both the quantum-layer operations and the main classical post-processing stages, emphasizing conceptual correctness rather than full implementation. In the quantum layer, raw bits are generated, encoded into qubit states, transmitted over a simulated quantum channel, and measured by the receiver using randomly selected bases. Following the quantum transmission, the protocol transitions to the classical post-processing phase, which includes basis reconciliation (sifting), parameter estimation through QBER calculation, information reconciliation to correct potential errors, and privacy amplification to remove any residual information that an adversary might obtain. To ensure protocol integrity, classical messages used during these steps are authenticated, and any optional experiment logs or configuration files are exchanged using safe, language-agnostic serialization formats (e.g., JSON), rather than executable serializers. This methodological structure allows the simulator to reflect the logical behavior of real BB84 implementations while remaining computationally accessible and suitable for educational and prototyping purposes. All simulations and analyses in this study were executed programmatically through Python scripts to ensure reproducibility and precise control over protocol parameters. Graphical user interfaces, while useful for instructional demonstrations, were intentionally excluded from the present analysis to maintain a clear focus on protocol-level behavior and statistical validation.

3.1 System overview and GUI

The simulator is implemented as a modular Python application with two user-facing execution modes: (i) a script-based mode for reproducible batch experiments and (ii) an optional Tkinter-based graphical user interface (GUI) for educational demonstrations. The GUI is designed to make the BB84 workflow observable by allowing users to configure the number of qubits (N), select the attack scenario (No-Eve vs. intercept–resend), and export experiment logs (e.g., bases, sifted indices, QBER results) for further analysis. Internally, the system follows a pipeline architecture: (1) random bit/basis generation, (2) circuit preparation and Aer execution, (3) measurement decoding, (4) sifting and QBER estimation, and (5) optional post-processing stubs (reconciliation and privacy amplification) for conceptual completeness. This separation between the interface layer (GUI) and protocol logic (modules) ensures that the educational interface does not alter the underlying BB84 computations and that identical results are obtained when running the same configuration in script mode.

3.2 Raw bit and basis generation

In the first stage of the BB84 protocol, both communicating parties independently generate the random values required for quantum state preparation and measurement. Alice begins by creating two sequences of length N: a raw bit sequence and a basis sequence. The raw bit sequence $A = \{a_1, a_2, \dots, a_N\}$consists of uniformly distributed binary values, while the basis sequence $B_A = \{b_{A1}, b_{A2}, \dots, b_{AN}\}$specifies whether each bit will be encoded in the computational basis (Z) or the Hadamard basis (X). These values are produced using a classical pseudo-random number generator, implemented through NumPy's random functions, ensuring that bit and basis choices follow the uniform distributions required by the BB84 protocol.

Simultaneously, Bob generates his own independent measurement basis sequence $B_B = \{b_{B1}, b_{B2}, \dots, b_{BN}\}$, where each basis is selected randomly between the Z and X bases. Bob's basis choices determine how incoming qubits will be measured and directly influence the subset of bits that will remain after the sifting stage. Because Alice and Bob generate their sequences independently and without prior coordination, the resulting basis mismatch naturally ensures that, on average, only half of the transmitted qubits will contribute to the sifted key. This randomness forms the foundational security mechanism of BB84, preventing an eavesdropper from predicting or aligning measurement bases.

3.3 Quantum state preparation and transmission

In the second stage of the protocol, Alice encodes each raw bit into a corresponding quantum state according to the basis she selected in the previous step. For each index i, the bit $a_i$and basis b_{Ai}jointly determine which qubit state is prepared. When the Z basis is chosen, Alice encodes the bit using the computational basis states $|0\rangle$and $|1\rangle$. When the X basis is chosen, she applies a Hadamard transformation to generate the diagonal basis states $|+\rangle = (|0\rangle+|1\rangle)/\sqrt{2}$ and $|-\rangle = (|0\rangle-|1\rangle)/\sqrt{2}$. This encoding ensures that any measurement performed in the incorrect basis results in an inherently probabilistic outcome, which is a fundamental source of security in the BB84 protocol.

The qubit preparation and transmission process is implemented using Qiskit's Aer simulator. For each qubit, the simulation initializes the state as $|0\rangle$. If the raw bit equals 1, an X gate is applied; if the chosen basis is X, a Hadamard gate is applied prior to transmission. Although this virtual environment does not involve a physical optical or fiber-based quantum channel, it accurately reproduces the mathematical behavior of qubit state preparation and basis-dependent encoding. The simulator executes the quantum circuit sequentially for all Nqubits, effectively modeling the one-way quantum communication channel from Alice to Bob. In this study, Aer is used in its ideal (noise-free) mode; therefore, the simulation does not approximate hardware decoherence unless an explicit noise channel (e.g., depolarizing, amplitude damping) is configured via Aer's noise model interface.

3.4 Measurement process

After the qubits are transmitted through the simulated quantum channel, Bob performs measurements on each

incoming qubit using the basis sequence he generated independently. For each index i, Bob measures the corresponding qubit in either the Z basis or the X basis depending on his chosen basis value b_{Bi}. If Bob selects the Z basis, he directly measures in the computational basis, yielding outcomes corresponding to $|0\rangle$ or $|1\rangle$. If Bob selects the X basis, a Hadamard gate is applied before measurement to rotate the qubit from the diagonal basis back into the computational basis, ensuring that the measurement produces results consistent with the BB84 protocol.

This measurement process is implemented using Qiskit's Aer qasm_simulator, which returns a classical outcome for each qubit. The measurement results form Bob's raw key candidate sequence $M_B = \{m_1, m_2, \ldots, m_N\}$. Because Bob's bases are chosen independently from Alice's, only the qubits measured in matching bases are expected to yield identical bit values. When the bases differ, the measurement outcomes become probabilistic, with a 50% chance of matching Alice's encoded bit. These mismatches naturally occurring or introduced by any eavesdropper serve as a fundamental indicator of channel disturbance and form the basis for the subsequent sifting and error estimation steps.

3.5 Sifting (basis reconciliation)

Following the measurement stage, Alice and Bob initiate the sifting process, in which they publicly exchange only their basis choices, not the bit values themselves. For each position i, they compare the basis used for encoding (b_{Ai}) with the basis used for measurement (b_{Bi}). If the two bases match, the corresponding bit is retained; if they differ, the bit is discarded. This step removes all positions in which Bob measured the qubit in an incompatible basis, leaving only the subset of measurements that should, under ideal noiseless conditions, match Alice's original raw bits.

The result of this procedure is the sifted key, defined as

$$K_s = \{a_i \mid b_{Ai} = b_{Bi}\}$$

On average, half of the transmitted qubits remain after sifting, since Alice and Bob select their bases independently and uniformly at random. This property is fundamental to the BB84 protocol: the randomness in basis selection ensures that neither an eavesdropper nor the legitimate parties can predetermine which positions will survive the sifting phase. Any disturbance introduced during transmission, whether due to malicious interception or due to noise in extended models, will manifest as errors within this sifted subset and will be quantified in the subsequent parameter estimation step.

3.6 Parameter estimation (QBER calculation)

Once the sifted key is obtained, Alice and Bob proceed with parameter estimation to determine whether the quantum channel has experienced any disturbance. This step is essential for detecting both measurement–disturbance effects and any potential eavesdropping attempts. To perform parameter estimation, Alice and Bob agree on a randomly selected subset of $m$positions from the sifted key. They publicly disclose their bit values only for these selected positions, allowing them to compute the Quantum Bit Error Rate (QBER).

The QBER is defined as the proportion of mismatches between Alice's and Bob's disclosed bits:

$$\text{QBER} = \frac{\text{Number of mismatched bits}}{m}.$$

In an ideal environment with no adversary, the expected QBER is approximately zero, as all surviving bits originate from matching basis choices. Under an intercept–resend attack, however, the expected QBER increases to roughly 25%, because an eavesdropper measuring qubits in random bases introduces systematic errors into the sifted key. This relationship forms a key security feature of the BB84 protocol: any unauthorized observation leads to detectable statistical deviations.

If the observed QBER exceeds a predefined threshold τ, the protocol is aborted, as it implies that the channel is compromised or exhibits excessive measurement–disturbance. Typical threshold values in practical BB84 implementations range between 11% and 14%, based on finite-key security analyses. Only when the QBER remains below this threshold do Alice and Bob proceed to the subsequent post-processing stages, which aim to correct remaining errors and eliminate any partial information that an adversary might have gained.

Additionally, the probability of detecting an intercept resend attacker when testing a subset of size m is given by

$$P_{\text{detect}} = 1 - (\frac{3}{4})^m.$$

describing the exponential decrease in Eve's chance of remaining undetected. In this study, the intercept–resend effect is treated as an information-theoretic disturbance arising from basis mismatch and state collapse under ideal measurements. It is not modeled as physical decoherence unless an explicit Aer noise model (e.g., depolarizing or amplitude-damping channels) is enabled. Therefore, the term "noise" in this paper refers exclusively to optional simulator noise channels, whereas "disturbance" refers to the BB84 measurement–disturbance mechanism under ideal quantum operations. Accordingly, claims about an 'error model' in this paper refer to the analytical error behavior of BB84 (e.g., the 25% QBER under intercept–resend) and not to hardware-level decoherence modeling.

3.7 Information reconciliation (error correction)

After sifting and QBER estimation, Alice and Bob may still have a small number of mismatched bits due to noise or transmission imperfections. To ensure that both parties share an identical intermediate key, they apply information reconciliation over the authenticated classical channel. Two widely used approaches exist: the interactive Cascade protocol and modern LDPC-based reconciliation. In the Cascade method, parity checks are exchanged over multiple passes to locate and correct erroneous bits, while LDPC-based methods use syndrome

vectors derived from sparse parity-check matrices to correct discrepancies more efficiently and with fewer communication rounds.

During the reconciliation process, certain parity bits or syndrome information must be revealed over the classical channel. This disclosed information represents a form of leakage to a potential adversary. The total leakage, commonly denoted as leak_{EC}, must be explicitly accounted for when determining the length of the final secret key. This ensures that the remaining key material preserves composable security and that any information potentially gained by an eavesdropper is fully removed in the privacy amplification stage.

3.8 Privacy amplification and classical channel authentication

After information reconciliation, Alice and Bob share an identical but not yet fully secure intermediate key. Although the reconciliation stage corrects errors, the information disclosed through parity checks or syndrome messages may provide an eavesdropper with partial knowledge about the key. To eliminate this residual information, the parties perform privacy amplification, a classical post-processing procedure that compresses the reconciled key into a shorter but information-theoretically secure final key. In this study, universal hashing is adopted as the conceptual model for privacy amplification, with Toeplitz-matrix-based hash functions serving as a standard and practical example. The final key length $\ell$is determined by the size of the reconciled key, the measured QBER, the amount of leakage during reconciliation, and the target security parameter ε, ensuring that the resulting key is secure in the composable security framework.

In addition to removing information gained by an eavesdropper, it is crucial that all classical messages exchanged during sifting, parameter estimation, and reconciliation are authenticated. Without authentication, an attacker could manipulate basis information, parity messages, or verification subsets, thereby compromising the integrity of the protocol even in the absence of direct quantum interception. To address this requirement, classical message authentication codes (MACs) based on universal hashing such as the Wegman–Carter scheme should be used. These methods provide information-theoretic authentication when combined with small amounts of pre-shared secret material.

Finally, all classical metadata exchanged between parties including basis lists, measurement outcomes, and reconciliation messages must be transferred using a secure and platform-independent format. In the revised framework, unsafe serialization mechanisms such as Python's pickle module are avoided due to their susceptibility to code-execution attacks. Instead, structured formats such as JSON or CBOR are recommended, as they ensure safe, transparent, and reproducible data exchange across different computing environments. In the implemented prototype, experiment artifacts are exported as non-executable structured files (JSON) with explicit schemas (e.g., fields for N, basis lists, sifted indices, and QBER), enabling integrity checks via deterministic parsing and optional hashing of the exported log. By integrating privacy amplification and authenticated communication, the simulated BB84 protocol adheres to modern QKD security standards and produces a final key that is provably secure under the assumed threat model.

4 Results

The results reported in this section correspond to an idealized analytical validation of the BB84 protocol in a software-only simulation setting. The reported sifted-key ratios, QBER behavior, and eavesdropper-detection probabilities are presented as theoretical expectations and ideal-simulator behavior under noise-free Aer execution, rather than as hardware-level measurements or performance benchmarks. Unless explicitly stated otherwise, the numerical values in this section are derived from closed-form BB84 analysis and are used to verify the internal consistency of the implemented workflow.

This section reports the key numerical outcomes obtained from the virtualized implementation of the BB84 protocol. The results focus on three fundamental performance indicators: (i) the proportion of raw bits that survive the basis-reconciliation process (sifted key rate), (ii) the quantum bit error rate (QBER) observed under both ideal and adversarial conditions, and (iii) the theoretical probability of detecting an intercept–resend eavesdropper based on the verification subset size. Together, these metrics provide a clear and structured assessment of how accurately the simulated environment reproduces the statistical behavior predicted by standard BB84 security analyses. By presenting the observed rates, error characteristics, and detection probabilities, this section establishes a quantitative foundation for evaluating the integrity and reliability of the virtualized system, which is further discussed and interpreted in Section 5.

4.1 Sifted key rate

In the BB84 protocol, only those qubits for which Alice and Bob select the same basis contribute to the final sifted key. Since both parties choose their encoding and measurement bases independently and uniformly at random, the probability of a basis match for any given qubit is $1/2$. As a result, approximately half of the raw bits transmitted through the simulated quantum channel are retained after sifting. This behavior matches the theoretical expectation: for N transmitted bits, the expected sifted-key length is approximately N/2, with deviations due to random basis selection. This outcome aligns precisely with theoretical predictions and demonstrates that the simulator correctly models the probabilistic structure of basis reconciliation in BB84. The sifted key serves as the input for the subsequent parameter-estimation and error-correction steps, and its size directly influences both the confidence level of eavesdropper detection and the achievable final secret-key rate. As such, the expected ≈50% survival ratio confirms that the implemented environment reflects the expected

operational characteristics of BB84 under ideal transmission conditions.

4.2 QBER in no-eve and eve scenarios

The quantum bit error rate (QBER) is the principal statistical indicator used to evaluate whether the quantum channel has experienced disturbance. In the absence of noise or adversarial intervention, BB84 predicts that qubits measured in matching bases should ideally yield identical outcomes. In an idealized noiseless setting, BB84 predicts a QBER close to zero when bases match and no adversary is present. To assess the simulator's response under adversarial conditions, an intercept–resend attack model was conceptually analyzed. In this attack, an eavesdropper measures each qubit in a randomly chosen basis and resends a new qubit based on her measurement outcome. Because Eve's basis selection matches Alice's only with probability $1/2$, and incorrect-basis measurements collapse the state unpredictably, the expected error rate introduced into the sifted key is approximately 25%. This theoretical value arises from the fact that Eve causes an error in the sifted positions with probability $1/4$, a property inherent to the BB84 protocol. Accordingly, under the standard intercept–resend model, the QBER in the sifted key is expected to approach ≈25%; therefore, for typical thresholds (τ≈11%–14%), the protocol would abort under this adversarial model.

By comparing the near-zero QBER in ideal conditions with the characteristic 25%error level expected under an intercept–resend attack, the results confirm that the virtualized implementation faithfully reproduces the statistical behavior essential for detecting channel disturbance in BB84. This clear separation between benign and adversarial scenarios forms the analytical basis for the eavesdropper-detection analysis presented in Section 4.3.

4.3 Eavesdropper detection probability

To quantitatively assess the ability of the protocol to detect an intercept–resend eavesdropper, we apply the standard BB84 detection model based on the verification subset revealed during parameter estimation. When an adversary measures each transmitted qubit in a randomly chosen basis, she introduces errors into the sifted key due to measurement–disturbance caused by basis mismatch prior to Bob's detection. For any disclosed verification bit, the probability that Eve's presence remains undetected is $3/4$: she chooses the correct basis with probability $1/2$, and when choosing the incorrect basis, her measurement coincidentally reproduces the correct bit with probability $1/2$. Consequently, the probability of missing Eve over a verification subset of size $m$is $(3/4)^m$, and the corresponding detection probability is:

$$P_{\text{detect}} = 1 - (\frac{3}{4})^m.$$

The detection probabilities reported in this section are obtained directly from the standard BB84 analytical model and are presented as a theoretical reference; no independent empirical or hardware-based validation of P_detect is claimed in this study.

This probability increases exponentially with the number of bits disclosed during parameter estimation. For example, when Alice and Bob reveal $m = 100$ sifted bits, the probability of detecting an intercept–resend attacker becomes:

$$P_{\text{detect}} = 1 - 0.75^{100} \approx 0.99999999999968.$$

This result illustrates that even a relatively small verification subset provides an overwhelmingly strong theoretical guarantee of detecting eavesdropping. The behavior is consistent with theoretical expectations for BB84 and confirms that the virtualized implementation conforms to the statistical detection characteristics underlying the protocol's security.

In contrast, when no eavesdropper is present and no simulator noise model is applied, the QBER in the verification subset remains close to zero, leading to $P_{\text{detect}} \approx 0$, as expected in an undisturbed channel. The sharp contrast between the ideal and adversarial cases (zero QBER versus approximately 25% QBER) validates the simulator's ability to reflect the detection mechanisms central to BB84 security.

To further illustrate the statistical strength of the BB84 eavesdropper-detection mechanism, Table 4 summarizes the detection probability as a function of the number of disclosed verification bits. The values are derived from the standard BB84 intercept–resend analysis and provide a theoretical baseline for assessing the confidence level of eavesdropper detection in the virtualized environment. The table highlights how rapidly the detection probability approaches unity as the verification subset size increases.

Table 4: Theoretical detection probability as a function of verification subset size

Verification bits disclosed (m)	**Theoretical P_{detect}**
10	0.9437
50	0.9999999
100	0.99999999999968
200	≈ 1.00

Detection probability as a function of the number of disclosed verification bits under an intercept–resend attack. Theoretical values are derived from the BB84 analytical expression $P_{\text{detect}} = 1 - (3/4)^m$.

5 Discussion

The results obtained from the virtualized BB84 implementation demonstrate that a software-only environment can accurately reproduce the fundamental statistical characteristics of quantum key distribution under idealized conditions. In particular, the expected sifted key rate of approximately 50% and the clear separation between near-zero QBER in undisturbed scenarios and the characteristic ≈25% QBER associated with an intercept resend attack are consistent with the

well-established theoretical behavior of the BB84 protocol [30]. These outcomes indicate that the simulator captures the basis-dependent measurement statistics underlying BB84, within an idealized setting without physical noise models. The following discussion first highlights the main contributions of the proposed framework, then outlines its inherent limitations, and finally identifies directions for future development. It is important to emphasize that the reported error behavior does not originate from physical noise or decoherence, but from the idealized measurement disturbance trade-off intrinsic to BB84. As such, the presented results should be interpreted as analytical validation rather than as a noise-performance evaluation.

Recent advances in quantum communication research indicate that practical QKD systems operate in noise-affected, lossy, and large-scale network environments. Comprehensive reviews of the field, such as the 2020 survey by Pirandola et al. [32], emphasize that state of the art QKD implementations rely not only on quantum channels but also on advanced post-processing techniques, including decoy-state protocols, finite-key security analysis, authenticated classical communication, and sophisticated error-correction schemes. While the present simulator reproduces the logical workflow and statistical behavior of BB84, it does not incorporate these full security layers. Consequently, the results should not be interpreted as evidence of composable security or as performance indicators for real-world QKD deployments, but rather as validation of analytical expectations in a controlled, idealized environment.

The contrast between ideal and adversarial QBER behavior predicted in this study is consistent with trends reported in experimental QKD research conducted between 2018 and 2024, including integrated-photonics-based systems, satellite-assisted key distribution, and metropolitan-scale quantum networks. Experimental studies have repeatedly shown that the introduction of realistic noise sources such as depolarization, phase drift, detector inefficiencies, and photon loss leads to deviations from ideal theoretical predictions. Incorporating such effects through noise models available in modern simulation frameworks would therefore constitute a natural extension of the present work and would enable more direct comparisons with contemporary experimental results.

The growing ecosystem of quantum-network simulation platforms provides additional context for positioning the contribution of this study. High-fidelity simulators such as NetSquid [33], SimulaQron [34], SeQUeNCe [23], and QuISP [35] are designed to model multi-node quantum networks, repeater architectures, and hardware-specific imperfections. In contrast, the simulator presented here prioritizes conceptual transparency and accessibility over full-stack physical realism. This design choice allows users to focus on the logical structure and statistical behavior of the BB84 protocol without the complexity associated with detailed hardware modeling, thereby complementing existing large-scale simulation tools rather than competing with them.

Although the foundational principles of QKD originate from early theoretical work [30,31], modern research has increasingly shifted toward scalable network integration, rigorous security proofs, and robust physical implementations. By situating the present results within this contemporary research landscape, the discussion clarifies that the primary contribution of this study lies in modeling the core quantum-measurement effects underlying BB84 namely sifting dynamics, QBER behavior, and eavesdropper detection via subset testing within an idealized, reproducible software environment. These features provide a reliable baseline for future extensions involving realistic noise channels, explicit privacy amplification and reconciliation modules, authenticated classical communication, and multi-node quantum-network simulations.

6 Conclusion

This study presented a software only implementation of the BB84 quantum key distribution protocol using Qiskit's Aer simulator, demonstrating that the essential statistical properties of quantum measurement and basis-dependent disturbance can be reproduced in an idealized virtual environment. The simulation correctly reflects the theoretical expectations of BB84, including a sifted-key rate of approximately half the raw key, near-zero QBER under noise-free operation, and the characteristic increase to around 25% QBER under an intercept resend eavesdropping scenario. These outcomes confirm that even a minimal simulation model can capture the fundamental mechanisms that enable eavesdropper detection in quantum key distribution.

The findings should be interpreted within the scope of an educational and prototyping framework rather than as a substitute for full-scale QKD implementations. The simulator does not yet incorporate critical components of practical QKD, such as realistic quantum-channel noise, authenticated classical communication, information reconciliation algorithms, or privacy amplification. Modern QKD systems rely on these advanced processes to achieve composable security under real world conditions, and integrating such mechanisms represents an important direction for future development.

Nevertheless, the results highlight the pedagogical value of accessible simulation platforms. By providing a controlled environment for observing core BB84 behaviors such as sifting dynamics, QBER trends, and the statistical basis of eavesdropper detection the simulator can support teaching, early-stage experimentation, and conceptual exploration. Future work will extend this framework to include Aer-based noise models, advanced post-processing modules, and eventually multi-node quantum-network simulations, enabling more comprehensive comparisons with contemporary QKD research and hardware implementations.

References

[1] Galindo, A., and Martín-Delgado, M. A. (2002). Information and computation: Classical and

quantum aspects. *Reviews of Modern Physics*, 347-423. Online first publication. http//doi.org/10.1103/RevModPhys.74.347

[2] Shannon, C. E. (1949). Communication theory of secrecy systems. *The Bell System Technical Journal, 28*(4), 656-715. Online first publication. http//doi.org/10.1002/j.1538-7305.1949.tb00928.x

[3] Çelik, S. (2021). Kuantum Kriptolojisi ve Siber Güvenlik. *Bilişim Teknolojileri Dergisi, 14*(1), 53-64. Online first publication. http//doi.org/10.17671/gazibtd.733309

[4] Hu, F., Lamata, L., Sanz, M., Chen, X., Chen, X., Wang, C., and Solano, E. (2020). Quantum computing cryptography: Finding cryptographic Boolean functions with quantum annealing by a 2000 qubit D-Wave quantum computer. *Physics Letters A*, 126214, Online first publication. http//doi.org/10.1016/j.physleta.2019.126214.

[5] Furrer, F. J. (2020). Roger A. Grimes. *Cryptography Apocalypse: Preparing for the Day When Quantum Computing Breaks Today's Crypto*. Springer.

[6] Clixto, M. (2009). Quantum computation and cryptography: An overview. *Natural Computing, 8*(4), 663. Online first publication. http//doi.org/10.1007/s11047-008-9094-8

[7] Gürsakal, N., and Çelik, S. (2020). *Kuantum Bilgisayarlar Teknolojik Anlamda Ne Getirecek? Küresel Ekonomiye Yön Veren Yeni Teknolojiler*. Ankara: Akçağ Yayınları.

[8] Feynman, R. P. (1982). Simulating physics with computers. *International Journal of Theoretical Physics, 2*1, 6-7. Online first publication. http//doi.org/10.1007/BF02650179

[9] Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. Quantum, 2, 79. Online first publication. doi:10.22331/q-2018-08-06-79

[10] Chen, L., Jordan, S., Liu, Y. K., Moody, D., Peralta, R., Perlner, R., and Smith-Tone, D. (2016). Report on post-quantum cryptography. *US Department of Commerce, National Institute of Standards and Technology*, 12, 1-8. Online first publication. http//doi.org/10.6028/NIST.IR.8105

[11] Microsoft Azure (2022). "Kübit nedir?", Received From: https://azure.microsoft.com/tr-tr/resources/cloud-computing-dictionary/what-is-a-qubit/#introduction

[12] Montanaro, A. (2016). Quantum algorithms: An overview. *npj Quantum Information*, *2*(1), 1-8. Online first publication. http//doi.org/10.1038/npjqi.2015.23.

[13] Shor, P. W., and Preskill, J. (2000). Simple proof of security of the BB84 quantum key distribution protocol. *Physical Review Letters, 85*(2), 441. Online first publication. http//doi.org/10.1103/PhysRevLett.85.441

[14] Grover, L. K. (1997). Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters, 79*(2), 325. Online first publication. http//doi.org/10.1103/PhysRevLett.79.325

[15] Hassan, N. A., and Hijazi, R. (2017). Data hiding techniques in Windows OS: A practical approach to investigation and defense. Syngress. Received From: https://www.researchgate.net/publication/317973198_Data_hiding_techniques_in_Windows_OS_A_Practical_approach_to_investigation_and_defense

[16] Şahin, F. (2015). Modern Blok Şifreleme Algoritmaları. *İstanbul Aydın Üniversitesi Dergisi, 7*(26), 23-40. Online first publication. http//doi.org/10.17932/IAU.IAUD.m.13091352.2015.7/26.15-21

[17] Toyran, M. (2007). Quantum Cryptography. *IEEE 15th Signal Processing and Communications Applications*, 1-4. Online first publication. http//doi.org/10.1109/SIU.2007.4298797

[18] Gümüş, E. (2011, February). Quantum cryptography and key distribution protocols. Paper presented at the Academic Computing Conference, İnönü University, Malatya. https://ab.org.tr/ab11/ozet/27.html

[19] Sasaki, M., Fujiwara, M., Ishizuka, H., Klaus, W., Wakui, K., Takeoka, M., and Zeilinger, A. (2011). Field test of quantum key distribution in the Tokyo QKD Network. *Optics Express, 19*(11), 10387-10409. Online first publication. http//doi.org/10.1364/OE.19.010387

[20] Peev, M., Pacher, C., Alléaume, R., Barreiro, C., Bouda, J., Boxleitner, W., and Zeilinger, A. (2009). The SECOQC quantum key distribution network in Vienna. New Journal of Physics, 11(7), 075001. Online first publication. http//doi.org/10.1088/1367-2630/11/7/075001

[21] Elliott, C., Colvin, A., Pearson, D., Pikalo, O., Schlafer, J., and Yeh, H. (2005). Current status of the DARPA quantum network. *In Quantum Information and Computation III, 5815*, 138-149. Online first publication. http//doi.org/10.48550/arXiv.quant-ph/0503058

[22] Stucki, D., Legre, M., Buntschu, F., Clausen, B., Felber, N., Gisin, N., ... and Zbinden, H. (2011). Long-term performance of the SwissQuantum quantum key distribution network in a field environment. *New Journal of Physics, 13*(12), 123001. Online first publication. http//doi.org/10.1088/1367-2630/13/12/123001

[23] Xu, F., Chen, W., Wang, S., Yin, Z., Zhang, Y., Liu, Y., ... and Guo, G. (2009). Field experiment on a robust hierarchical metropolitan quantum cryptography network. *Chinese Science Bulletin, 54*(17), 2991-2997. Online first publication. http//doi.org/10.48550/arXiv.0906.3576

[24] Hughes, R. J., Nordholt, J. E., McCabe, K. P., Newell, R. T., Peterson, C. G., and Somma, R. D. (2013). Network-Centric Quantum Communications. *In Frontiers in Optics 2013. Online first publication*. doi: https://doi.org/10.48550/arXiv.1305.0305

[25] Koushik, C. S. N., Choubey, S. B., Choubey, A., and Pachori, K. (2020). A Literature Review on Quantum Experiments at Space Scale—QUESS Satellite. *Innovations in Electronics and Communication Engineering*, 13-25.

[26] Cross, A. (2018). The IBM Q experience and QISKit open-source quantum computing software. *APS 2018*, L58-003.

[27] Shor, P. W. (1994). *Algorithms for quantum computation: Discrete logarithms and factoring*. Proceedings 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 124-134. Online first publication. http//doi.org/10.1109/SFCS.1994.365700

[29] IBM (2018). Kuantum Devreleri İçin Açık Bir Yüksek Performans Simülatörü, IBM Araştırma Yazı İşleri Ekibi. Received From: https://qiskit.org/documentation/tutorials/simulators/1_aer_provider.html

[30] Bennett, C. H., and Brassard, G. (1984). Quantum cryptography: Public key distribution and coin tossing. *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, Bangalore, India, 175–179. https://doi.org/10.48550/arXiv.2003.06557

[31] Ekert, A. K. (1991). Quantum cryptography based on Bell's theorem. *Physical Review Letters, 67*(6), 661–663. https://doi.org/10.1103/PhysRevLett.67.661

[32] Pirandola, S., Andersen, U. L., Banchi, L., Berta, M., Bunandar, D., Colbeck, R., et al. (2020). Advances in quantum cryptography. *Advances in Optics and Photonics*, 12(4), 1012–1236. https://doi.org/10.1364/AOP.361502

[33] Coopmans, T., Knegjens, R., Dahlberg, A., Maier, D., van Dam, S., and Wehner, S. (2020). NetSquid: A NETwork Simulator for QUantum Information using Discrete events. *Quantum Science and Technology*, 5(2), 025003 https://doi.org/10.1088/2058-9565/ab6bf9

[34] Dahlberg, A., Skrzypczyk, M., Coopmans, T., and Wehner, S. (2018). SimulaQron: An emulator for developing quantum internet software. *npj Quantum Information*, 4, 29. https://doi.org/10.1038/s41534-018-0070-7

[35] Aguado, A., Kraft, M., van der Meer, R., and Wehner, S. (2021–2024). QuISP: A Quantum Internet Simulation Package. Software Repository: https://github.com/sfc-aqua/quisp