

Enhanced Cryptography through Quantum Key Distribution and Satellites: The BB84 Protocol

Xiaoya Gao, Avni Iyer

Abstract

Today, government and military units rely on satellites to transmit sensitive data. However, the rise in computational power of quantum computers poses a significant threat to traditional cybersecurity algorithms, rendering them highly susceptible to attacks. Quantum Key Distribution (QKD) offers a promising solution by using principles of quantum mechanics such as superposition and the uncertainty principle to eliminate mathematical algorithms, thus increasing eavesdropping detection accuracy drastically. Utilizing QKD for satellites in Low Earth Orbit is key for the future of our cybersecurity as they increase efficiency by eliminating fiber repeaters and establishing multiple connections through a single satellite. The BB84 protocol is a QKD single-photon based algorithm that solves the intercept-and-resend attacks that are commonly employed by hackers. Through this study, we use three Python 3.12 simulations to explore the shortcomings of current BB84 protocol approaches for satellite applications. Furthermore, we illustrate why privacy amplification and error channel connections are essential for total cybersecurity.

Introduction

Secure networks are important for both day to day life as well the transmission of government and military information. Unfortunately, cyber attacks have increased by 600% since the pandemic, making enhanced cryptographic techniques a priority.

Traditional cryptography, which uses keys to encrypt and decrypt sensitive data, relies on mathematical algorithms. The data is transformed into a cipher text that is meant to be unreadable by a third party hacker. The method used to encrypt the data through a key relies on the difficulty in factoring large prime numbers. However, with the rise of quantum computing, this factoring method is easily cracked through Shor's and Grover's algorithms which employ quantum mechanics principles such as quantum parallelism and quantum Fourier transform.

In comparison to classical bits that take the form of either a 0 or 1, quantum computers use qubits that are a superposition of both 0 and 1. Thus, quantum qubits, which are in the form of photons, can hold significantly more information. Because photons are quantum particles, they also obey other laws of quantum mechanics. Superposition and quantum entanglement are what allow for quantum computers to perform operations so quickly: the computer can carry out multiple operations at the same time. The number of bits required to brute force an algorithm is significantly decreased, often by factors of 2. Scientists estimate that a classical computer would take 18 years to crack the AES-256 algorithm, by which the data would render useless, but a quantum computer could crack the algorithm within hours or even minutes by 2050 (figure 1).

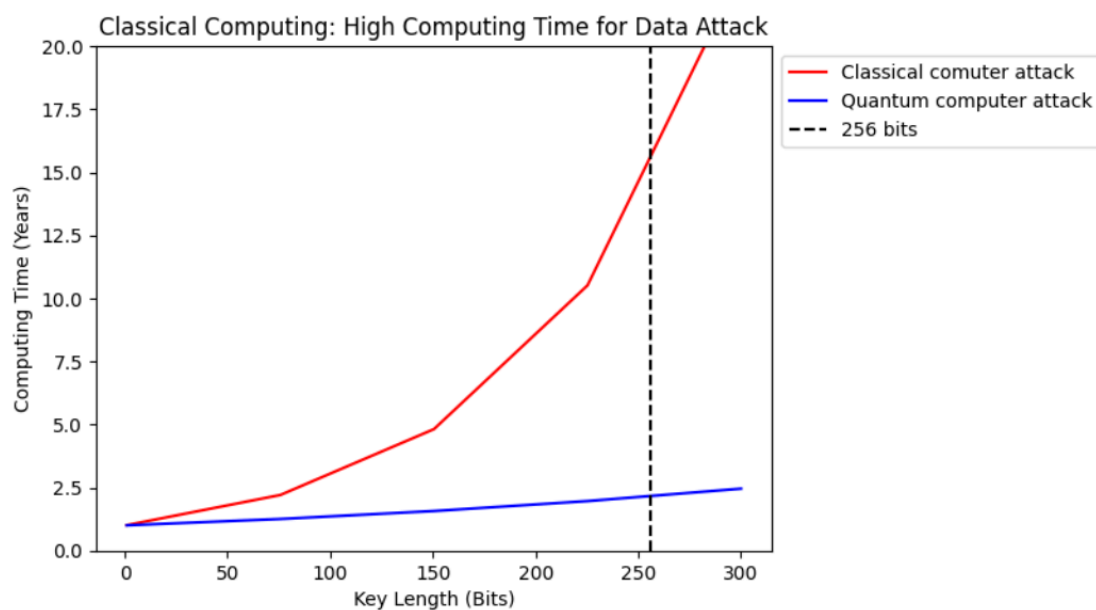


Figure 1: While a classical computer would take around 18 years to break AES-256, a quantum computer will be able to in hours (author).

To combat this threat, Quantum Key Distribution (QKD) utilizes the very properties of photons that quantum computers take advantage of to expose an eavesdropper. However, QKD is not without flaws. This paper specifically focuses on the shortcomings of the BB84 Protocol and the future directions of QKD as an integral part of cryptography.

Background

Traditional Cybersecurity and the Quantum Threat

In our current day to day use, cybersecurity of communication are amplified by methods of traditional cryptography that utilizes mathematical operations to encode text and information into long strings of cipher text. A traditional AES-256 encryption algorithm, which uses 256 bits of encryption key, is already considered unbreakable and widely used by the government and military. With the rise in computational power of quantum computers, however, everything could change. The properties of quantum mechanics significantly enhance the speed at which computers could do operations: the superposition principle allows the quantum particles running the computer to be in many different states at once, and the computer could withstand running up to millions of operations at the same time, greatly out-powering the best of traditional computers. Scientists estimate that by 2050, quantum computers will be powerful enough to decrypt AES-256 encryption within minutes, leaving our future cybersecurity vulnerable for attacks.

The BB84 Protocol

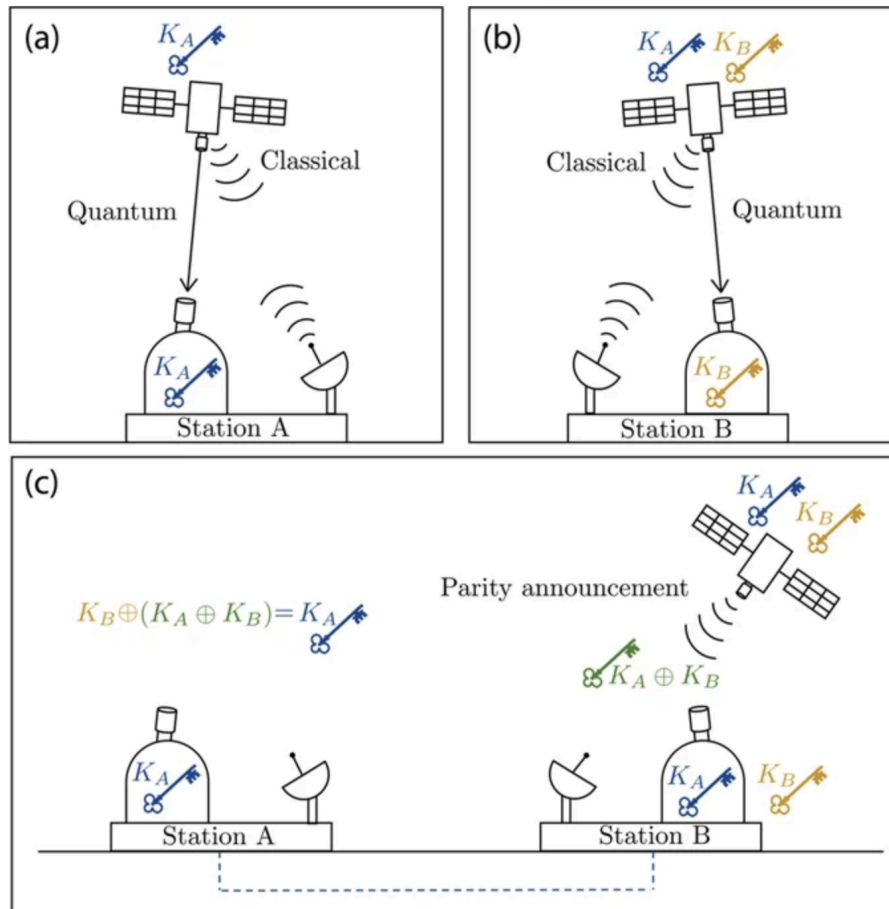
Satellites Application

In modern day, governments and military alike already utilize satellites to communicate and transfer sensitive data and top secret information. The advantages that satellites have over other methods in communication is already globally recognized, prompting more reliance on satellite communication. This poses a problem in the future, when the mathematical encryption used by satellites can be easily broken, opening a gap for eavesdroppers.

Satellite application of the BB84 Protocol is crucial for our future of cybersecurity and SATCOM, where it removes the mathematical barrier of encryption so there really is no way to decrypt the ciphertext without the decryption key. Satellites offer a better infrastructure setup: removing the need of fiber repeaters which lowers cost and security risks.

The terrestrial application of the BB84 Protocol requires the practice of fiber optics, where photons travel through a thin strip of material with high refractive index by total internal reflection, thus requiring miles of fiber repeater cables to transmit the photons. This provokes a security risk where anyone can physically destroy the cables in the middle, stopping the transmission of photons completely, and increasing the cost needed to replace the damaged cables. Moreover, a tremendous amount of fiber repeater cables must be used to connect different stations to each other, and the number only grows when the number of stations increase, posing a

great cost problem for governments. Satellites offer a promising solution by being able to establish multiple connections through a single satellite and sending down photons in a beam to the stations from outer space. Stations on the ground require a photon receiver and a classical channel connected with the satellite.



The satellite first communicates with Station A. Using the BB84 Protocol, they form a key, denoted by K_A . The satellite then repeats this process with Station B, creating another key K_B . Notice now the satellite possesses both keys, while each grounded station only possesses their own respective key.

Let's define $K_A = 01001111$, $K_B = 10010101$, which are randomly chosen 8-bit binary strings. $K_A \oplus K_B = 01001111 \oplus 10010101 = 11011010$, and $K_B \oplus (K_A \oplus K_B) = 10010101 \oplus 11011010 = 01001111 = K_A$, so we conclude that $K_B \oplus (K_A \oplus K_B) = K_A$, and this works for any string of binary bits of any length. (Note: \oplus denotes the XOR logic gate for binary bits.)

With this logic operation, the satellite sends down to station B the results of $K_A \oplus K_B$ through the classical channel, and since station B knows the key K_B , K_A can be easily calculated. This process is repeated with Station A, and now both stations know both of the keys, and an eavesdropper has no chance of figuring them out since the keys sent through the classical channel are not the encryption nor the decryption keys. Now, when the ciphertext is sent to station A and B using the respective encryption keys, they can use the other key for decryption and access the information that had been encrypted.

Satellite Challenges

Although applying the BB84 protocol to satellites seems like a well-rounded solution for our future cybersecurity, there are challenges we will inevitably face. Our current satellites that are in orbit do not have the primary function for quantum key distribution, and any photons that are sent down to the Earth have a relatively high chance for channel errors and crude light bouncing. Our technology for receiving the photons are not advanced enough either to fully carry out satellite quantum communication. Moreover, atmospheric changes easily affect QKD, since it purely relies on the physical state of the photons. Natural phenomena such as clouds and rain can affect how the photons are received by the stations.

Error Correction

In application, quantum channels are noisy. This noise can occur because of infrastructure factors such as birefringence due to optical fibers, mistakes in photon preparation, and dark counts by photon detectors; even environmental interference due to heat can introduce error because of the sensitive nature of photons.

Performed after key sifting and in a classical channel, error correction is essential to prevent mistakes that occur due to this background noise. This means that the amount of key that is usable after error correction reduces for two reasons: error correction along the classical channel introduces bit flips and allows the eavesdropper to access information. Based on the Shannon capacity, the amount of the key that is usable after error correction is described by the following equation:

$$R_k = R_s (1 - 2 h_2(p_e))$$

Where R_k is the useful key, R_s is the sifted key, h_2 is the binary entropy function, and p_e is the error rate due to background noise. The binary entropy function describes the amount of information Eve has access to. The binary entropy formula is as follows:

$$h_2 = -[(1-p) * \log_2(1-p) + p * \log_2(p)]$$

These formulas show that once the p_e reaches 11%, the key length is reduced to 0 and becomes completely unusable.

Error correction methods vary greatly. One example is Cascade, formulated by Brassard. This method, used in QKD, continues to divide the key into smaller and smaller blocks while performing parity checks until the parity error is identified. However, this requires a lot of information to be shared between the sender and receiver, increasing cost and time. Other methods, such as Reed-Solomon and Turbo Codes, are difficult to implement in QKD. Thus, the method used in this paper is Low-Density Parity Check (LDPC).

LDPC uses G , a matrix encoder, to create H , a parity-check matrix with redundant parity bits. This matrix has the shape $m \times n$. m represents the number of parity checks, while n demonstrates the number of bits in the codeword. The following parity-check equation represents an LDPC error correction iteration:

$$\bar{x}H \pmod{2} = \bar{0}$$

In this case, \bar{x} is the column vector containing the message plus redundant parity bits. Breaking this equation down further necessitates the understanding of a few key concepts:

Redundant parity bits are bits added to a message that both the receiver and sender are aware of. These help in checking for errors.

H is the parity-check matrix. Each row in H is multiplied by \bar{x} and must be 0. Thus, LDPC iterates through parity checks to ensure that the *parity vector* -- the product -- is 0.

Thus, the receiver can be certain that both H and the parity vector are fixed values: the keystone of error correction. During the decoding process, a similar equation is employed:

$$\bar{p} = H\bar{y} \pmod{2}$$

A threshold for the number of times a message bit fails a parity check is mathematically computed before the bit is flipped. Then, the equation iterates until \bar{p} reaches $\bar{0}$. Sometimes, however, LDPC fails to converge after which point failure is declared.

Privacy Amplification

Privacy amplification is meant to further limit the amount of information Eve has access to. The length of the key is shortened using hashing methodologies; the length is determined by the ratio R_k/R_s that depends on the amount of error present in the channel.

Methods

Using Python 3.12, we conducted a study to demonstrate and evaluate the significance of LDPC error correction methods for a BB84 protocol simulation. The study consisted of 3 trials

- 1) Simple BB84 protocol without eavesdropping or error correction
- 2) BB84 protocol key exchange with eavesdropping but without error correction
- 3) BB84 protocol with error correction but no eavesdropping.

For all three simulations, a background error rate of 0.5% was employed. This error rate mimics a sophisticated QKD infrastructure set up used in highly sensitive data exchange. A more rudimentary QKD setup would likely contain a higher rate such as 3%.

The code used in this study is available: <https://github.com/avniiyer/Enhanced-Cryptography-through-Quantum-Key-Distribution-and-Satellites-The-BB84-Protocol>

First Simulation

To simulate the BB84 protocol, methods such as superposition and entanglement are defined as functions. First, Alice's qubits and basis are randomized. Then, for photons polarized using the diagonal basis, a quantum gate is applied to transform the initial vector into a superposition state. This quantum gate is called Hadamard's matrix, where the matrix is represented as $H = 1 / \sqrt{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. Applied to the initial eigenstate, the resulting matrix is as shown:

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

Quantum entanglement is the phenomenon in which subatomic particles remain linked despite distance. This quantum entanglement is simulated in the BB84 Protocol as well through the use of the Bell State:

```

state_00 = np.array([[1], [0], [0], [0]])
# |11> state
state_11 = np.array([[0], [0], [0], [1]])
# Bell state ( $|00\rangle + |11\rangle$ )/sqrt(2)
entangled_state = (1/np.sqrt(2)) * (state_00 + state_11)
return entangled_state

```

Quantum entanglement is used to link Alice's and Bob's photons and distinguish from Eve's in an ideal QKD transmission.

Finally, Bob measures the qubits in a similar but reverse process. When Bob measures the photons, his basis are randomized. If the basis were diagonal, the qubits are taken from a superposition state to a collapsed state (see Simulation 2 for a more detailed explanation) and thus are prone to error because there is a 50% chance of Bob guessing incorrectly.

One sample result of this first simulation is shown below.

```

Alice's bits:  [0 0 1 1 0 0 0 1 1 0 1 0 1 1 1
1 1 0 1 1 1 1 0 1 1 0 0 1 0 0 1 1 0 1 1 0 0
0 1 0 0 0 1 0 0 0 0 1 1 1 1 0 1 1 1 1 1 0 0
0 0 0 1 1 0 1 1 0 1 1 1 1 1 0 0 0 1 0 1 0 1 1
1 0 0 0 1 0 1 1 0 1 0 0 1 1 0 0 0 1 0 0 0 0
1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 0 1
1 1 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 0
Length of Alice's key: 256

Bob's results: [0, 1, 1, 0, 0, 0, 0, 1, 1, 0
Length of Bob's key: 256

Simulated BB84 Key:
[0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0,
Final Key:
[0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0,
shared key: 125 out of 256
Error rate: 0.80%

```

Figure : The shared key, or the final key, is shorter than both Bob and Alice's original keys because only shared measurements are kept.

The error rate is due to the random error of 0.5% used to simulate slightly noisy quantum channels.

Second Simulation

Implementing eavesdropping involves collapsing the quantum states before the qubits reach Bob. Qubit states are given by measurement operators. The probability that Eve measures a 0 or a 1 is given by the inner product of the state vector (wave function) with a measurement operator. Based on a randomly generated number, Eve's probability of measuring a 0 or a 1 is calculated if the number is less than or greater than the probability of measuring 0. Either way, the state has collapsed and this is what Bob measures at the end of the transmission.

Thus, when performing the second simulation with eavesdropping, the following code was used to demonstrate the collapsed state due to Eve's eavesdropping:

```
prob_0 = (state.conj().T @ M_0 @ state).item()
prob_1 = (state.conj().T @ M_1 @ state).item()
# Collapse state based on measurement outcome
if np.random.rand() < prob_0:
    collapsed_state = M_0 @ state / np.sqrt(prob_0)
    measurement = 0
else:
    collapsed_state = M_1 @ state / np.sqrt(prob_1)
    measurement = 1
```

Aside from eavesdropping, this simulation runs very similarly to the first simulation, with entanglement and superposition generated from Alice's qubits and Bob measuring the qubits at the end. A sample result of running this simulation once is shown below:

```

Alice's bits: [0 0 1 1 0 0 0 0 0 1 1 0 1 0 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 1 1
1 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1
0 0 0 1 1 1 1 1 1 0 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 0 0 1 1 1 0 0 1 0
0 0 0 1 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 1 1 1 0 1 0 0 0
0 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1
1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 0 1 1 0 1 0 1 0 0 1 1 0 1 1 0 1 0 0 0 0 0 1
0 0 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 1 1 0 1 1 1 0 0 0 1 1 0 1 1 1 0 0]
Length of Alice's key: 256

Bob's results: [0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1,
Length of Bob's key: 256

Simulated BB84 Key:
[0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1,
Final Key:
[0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1,
shared key: 120 out of 256
Error rate: 3.33%

```

Figure x: Similar to the first simulation, the resulting sifted key is shorter than both Bob and Alice's.

This time, the error rate is due to both random background error and eavesdropping error.

Third Simulation

The third simulation employs LDPC: a powerful error correction method. First, the library from `pyldpc` was imported to get access to `make_ldpc`, `encode`, `decode`, and `get_message`.

The encoding matrix, G , was created with redundant parity bits in order to form the codeword based on Alice's bits. After random bits were flipped based on the error rate, a "decode" function used the parity check matrix, H , to iterate through the message and check if the parity vector is 0. This is given by the following equation:

$$H\mathbf{c}^T = \mathbf{p}$$

Where H is the parity matrix, \mathbf{c}^T is the transpose of the code word, and \mathbf{p} is 0 if the error is 0. The LDPC code does not always converge to 0 but it is designed to get close.

Next, by importing sha256 from the library hashlib, the key is converted into a hash (an example is shown below). This reduces the amount of information Eve can understand. This process is known as privacy amplification.

Final Key:

6d86701d08b4ceab2f98595358c31705ba0231cd592d865d4cf02095b3d9cc05

Fig x: the final key is simply a has of the shared key using privacy amplification.

Results and discussion

To create more generalizable conclusions, both simulation 1 and simulation 2 were run 20 times. The error (the number of differences between Bob's and Alice's qubits when the bases were matching) are given for each trial.

Trial	Without Eve (%)	With Eve (%)
1	4.35	4.51
2	5.93	8.21
3	8.27	5.26
4	3.88	3.54
5	6.72	7.35
6	4.10	5.34
7	4.38	7.26
8	9.16	7.14
9	6.34	4.41
10	5.34	4.83
11	7.09	7.80
12	3.79	7.14
13	4.76	4.35
14	7.56	7.89
15	4.84	4.84
16	8.00	5.60
17	2.88	5.00
18	6.35	4.38
19	2.56	4.96
20	4.51	3.33
	Without Eve	With Eve
Averages (%)	5.54	5.66

Figure x: A simple BB84 Protocol without error correction for both without eavesdropping and with eavesdropping was performed 20 times with the averages shown.

As shown, the average error between the BB84 protocol without and with eavesdropping only had a difference of 0.12%. This poses the question of how to distinguish between eavesdropping error and background error when the difference in error rate is so small.

This is where LDPC error correction methods come in. As discussed under Methods, LDPC cycled through the message, encoding it using the G matrix into an H parity check matrix and tried to converge to 0. While it did not always converge to 0% error, a majority of the runs were able to reduce the error to 0% and therefore check back against random channel error.



Simulation Number	Eavesdropping simulated?	Error correction method (LDPC)?	Error (mismatch between Alice and Bob's bits)
1			5.54%
2			5.66%
3			0.00%

Fig x: In the first simulation without eavesdropping or error correction, the average key error was 5.54%. In the second simulation with eavesdropping but without error correction, the average key error was 5.66%. In the final simulation without eavesdropping but with error correction, the error was able to reach 0.00%.

It is important to note that this simulation is not completely representative of a real quantum channel because there are a lot more factors contributing to background error and it is unlikely the LDPC will converge to a flat 0.00% often. However, these simulations serve to show the importance of using error correction to reduce background error noise in order to highlight the error an eavesdropper would bring in. Without error correction, this distinction would be impossible. While the third simulation also included a demonstration of privacy amplification, there was no way to measure the success of privacy amplification due to a lack of actual messenger and receivers. However, Watanabe et. al finds that privacy amplification increases the error threshold even further, by about 3.5%.

Using satellites in conjunction with error correction and privacy amplification methods for Quantum Key Distribution is another step towards reducing background noise as they reduce the need for fiber repeaters, which are susceptible to tampering and environmental interference. However, there is still work to be done. A major source of error is atmospheric interference that

disturbs sensitive photons. One possible future solution for this type of error is pilot qubits (qubits known by both Alice and Bob prior to communication) that can show Bob the systematic error of the photons caused by the atmosphere. With this knowledge, Bob can then compensate for the error of the actual message. Using pilot qubits in a similar computer simulation is an important area for future research.

Conclusion

Through the use of python simulations and a thorough literature review, it is evident that error correction methods are a necessity in secure quantum key distribution algorithms in the future. Future research into more novel error correction methods beyond LDPC would be useful. With regards to satellite communications, the use of pilot qubits is an important field of research to compensate for the error brought in by atmospheric changes.

References

Hasnainistz. Quantum Computing: Why it is so fast? - Hasnainistz - Medium. Medium. December 11, 2023. <https://medium.com/@hasnainistz/quantum-computing-why-it-is-so-fast-ddc93f3dffd#:~:text=Superposition%20and%20Entanglement%3A%20The%20Speed,compared%20to%20their%20classical%20counterparts.>

Durbha, Subramanyam. Cryptography and factorization methods in cryptography. Diss. Rutgers University-Camden Graduate School, 2018.

Valle, Chris. Shor's Algorithm and Grover's Algorithm in Quantum Computing. Diss. University of Kansas, 2011.

Gent, E. *Quantum computers could crack encryption sooner than expected with new algorithm*. Singularity Hub. <https://singularityhub.com/2023/10/02/quantum-computers-could-crack-encryption-sooner-than-expected-with-new-algorithm/#:~:text=Most%20implementations%20of%20RSA%20rely,crack%20a%20number%20that%20large.>

Chan, Philip. Low-density parity-check codes for quantum key distribution. *Ottawa : Library and Archives Canada* 2010. 731320310. <https://library-archives.canada.ca/eng/services/services-libraries/theses/Pages/item.aspx?idNumber=731320310>

Acknowledgements

Author Information