Project title: Simulation of Attacks on Quantum Key Distribution protocols

(Discussion on cryptography in the era of Quantum Computers and simulation of attacks on QKD protocols using Qiskit)

Course: Quantum computation: Introduction to algorithms and implementation using Qiskit 2022

Teachers-in-charge: Dr. Chetan Waghela and Dr. Dintomon Joy

(submitted by)

Supreeth BS • Aritra Mukhopadhyay • Sagnik Sinha Choudhuri
Atanu Singha ● Kirty Ranjan Sahoo • Sagar Gaur • Soniya • Samyak Parashar •Abeena U
● Katha Haldar

CONTENTS

Introduction to cryptography

Key distribution is the most crucial facet of cryptography. To understand it, we first begin by an introduction to cryptography and the processes involved in communication.

The following are the steps involved in the process of communication:

Information source → source compression → channel encoding → encryption → decryption → channel decoding → source decompression → information

Cryptography concentrates on encryption and decryption. The first question we ask is what is the need for encryption and decryption? To answer this, let's take an example. Suppose Alice wants to share a one-time password '010' to Bob, she first encodes this as part of the error correction process using a repetition code 000111000, this is done because the channel may introduce some errors such as bit-flips or erasures. If she sent 010 and the channel introduced a bit-flip to the last bit, the message is changed to 011 which is not what she intended to communicate to Bob. To avoid this, she uses a 3-bit repetition code and encodes the message 010 to 000111000; this is the channel-encoded message. Even if the channel introduced bit-flips, Bob can use a majority vote decoder to decode the message correctly.

However, it doesn't suffice to just send the channel-encoded message. Not only are we required to protect the message from errors introduced by the channel, but we are also required to protect the message from falling into the hands of an eavesdropper who goes by the name Eve. If Alice just tried to send 000111000 to Bob, and it fell into the hands of Eve. She could, in principle, understand that Alice was using a repetition code to send the message 010 and easily misuse the obtained information and therefore indulge in unethical means.

In order to protect the message from Eve, we need to convert the message into a form that cannot be understood by Eve even if she accessed it. It is important to note that the access to the message may not be prevented, but surely it can be ensured that the accessed message makes no sense to Eve thus rendering it useless for exploitation.

This process of converting a meaningful message into what seems like gibberish to Eve is called Encryption and this happens at Alice's end similarly, the process of converting the received gibberish into a meaningful message is called decryption. And the practice and study of techniques associated with encryption and decryption is called Cryptography.

All of this can be thought of as a lock and key mechanism. To encrypt the message, Alice uses a key to lock the message. Now that the message is encrypted/locked, it is secure. To decrypt the encrypted/locked message, Bob also uses a key. Depending on whether Alice and bob use the same key or different key, Cryptography is classified into two different types, namely symmetric cryptography (private key cryptography) and Asymmetric key cryptography (public key cryptography)

Symmetric cryptography

In symmetric cryptosystems, both Alice and Bob use the same cryptographic keys for both encryption and decryption. This can be thought of as a safe, where the message is locked by Alice with a key. Bob in turn uses a copy of this key to unlock the safe. If Alice wants to send a message 000111000 to Bob, she uses a randomly generated key e.g. k=110010100. A copy of this key has to be distributed to Bob so that he could use this for unlocking. Now, Alice simply adds each bit of the message with the corresponding bit of the randomly generated key to obtain S=000111000+110010100=110101100 (binary addition modulo 2 without carry). This is the locked message or encrypted message or gibberish. It is then sent to Bob, who decrypts the message by subtracting the key and obtains m = s - k = 110101100 - 110010100 = 000111000, which is the desired message. Upon successful distribution of the copy of the key, it is ensured that only Alice and Bob know the keys and the whole process is as safe as anything could be. Although perfectly secure, the problem is that it is essential for Alice and Bob to possess a common secret key; this cannot happen without the prior distribution of the key. During the cold war, the communication between Russia and America was happening with a prior distribution of the key through the embassies as they could not risk sharing it through an unsecure channel. Implementing secure communication for the masses using this technology is economically not viable. Because of this reason, a new type of cryptography called asymmetric cryptography was proposed which still exists to this day.

Asymmetric cryptography

Here, Alice and Bob use different keys for encryption and decryption. This pair consists of a public key (which may be known to others) and a private key (which may not be known by anyone except the owner). The first implementation was done using the RSA and is still widely used. We will show an example of how the rsa works to better understand this scheme

Say Alice wants to send some message to Bob. To do this securely, Bob prepares some public and private keys. The public key allows anyone to send him an encrypted message (i.e ciphertext) and private key allows only Bob to decrypt the ciphertext into meaningful plaintext. To do this, Bob begins by choosing two distinct prime numbers ‘p’ and ‘q’. For example, say p = 17 and q = 41. Bob keeps these two numbers secret but reveals their product,n=p×q=17×41=697 and sends it to alice.Bob then computes the φ = (p−1)×(q−1)=16 ×40 = 640, which Bob keeps secret. Then he tries to find an integer e such that 1 < e < φ and e is coprime to φ, meaning e and φ have greatest common divisor equal to 1 which can be written as gcd(e, φ)=1. We can use Euclid’s algorithm to find an e such that gcd(e,φ) = 1.Bob has to select one of the many values of ‘e’ which satisfy above mentioned conditions.
Suppose Bob selects e=3 which is less that φ and co-prime of φ. Now, Bob publishes the numbers ‘e’ which can be used as a public key by Alice to send a message to Bob.Finally, Bob computes $d = e^{-1} mod\ \varphi = 3^{-1} mod\, 640 = 427$,where the modulo operation(mod) returns the remainder of a division example q mod w returns the remainder which we get while dividing q by w. Here d is modular multiplicative inverse of $e\ mod\ \varphi$,From number theory which always exist as e was chosen coprime to φ .Hence, We should get 1 as a result of $(d \times e\, mod\, \varphi) mod\, \varphi$ =
$((e^{-1} mod\ \varphi) \times (e\, mod\, \varphi)) mod\, \varphi$
=$(e^{-1} \times e) mod\, \varphi$ (by Using properties of modulo operation)
=$1\, mod\, \varphi$ = 1

The number d is Bob’s private key, so he keeps it a secret. ‘d’ will allow him to decrypt messages sent to him. Now, let’s see how public key ‘e’ and ‘n' are used to encrypt the message and private key ‘d’ and 'n' are used to decrypt it. Say Alice has to send A=2 as a message to Bob.She computes the ciphertext C = $A^{e} mod\, n$= $2^{3} mod\, 697 = 8$ and she sends it to Bob.When Bob receives the ciphertext C, he computes $C^{d} mod\, n = A^{ed} mod\, n = A^{1} mod\, n = A$, receiving the message A.

Continuing our example, Bob computes $8^{427} mod\, 697 = 2$,which is the message Alice wants to send to Bob.
Decryption is hard for an eavesdropper(Eve) because she does not know the secret key d. She only knows Bob’s public keys, n and e. To find the secret key $d = e^{-1} mod\, φ$ Eve needs to know φ = (p−1)(q−1), but this requires knowing p and q, which involves factoring n. If we choose p and q to be large prime numbers, then there is no known efficient, classical algorithm for factoring large numbers, although a quantum computer can efficiently factor numbers using the Shor’s algorithm.

Quantum Computing: A threat to asymmetric cryptography

If we have a large number (N) which is the product of two prime numbers, factoring such a number to find the original two numbers is a tedious guess even for the best of classical supercomputers. As the size of the number (N) gets bigger it reaches a point where the time required by any currently existing classical algorithm to find such factors is unreal. This fact is employed in the RSA encryption protocol described above to encrypt data today in such a way that decrypting the data requires prime factors of the large number.

Our current best method to find such factors is essentially to guess a number that might be a factor and then check if it is indeed a factor, and if it isn’t then, to repeat the process until we find the factor. This is called the Brute-force method. There are clever ways to make better guesses, however even with those, the process is simply too slow. Hence a sufficiently large number (N) becomes impossible to factor in any reasonable amount of time.

Shor’s algorithm (using quantum superposition and entanglement) exponentially decreases the time required to find the factor of large numbers. Roughly speaking, the Shor’s algorithm starts with a random guess that might share a factor with the target number and then the algorithm transforms it into a much better guess that has a considerably higher probability of sharing a factor with the target number and the process is repeated with better guesses until the required factor is found.

It can be run on a classical computer as well, however it is the process of turning a bad guess into a better one that requires a lot of processing power.

We start with a number N which is a product of two unknown numbers and make a guess g which might share a factor with N. Now for any pair of whole (here g,N)
$g^{p} = mN + 1$ for some number p and m

$g^p - 1 = mN$
$(g^{p/2} + 1)(g^{p/2} - 1) = mN$

So, now we have two factors of N. As simple as this process might appear there are a few problems and the most prominent one is how to find p. Trying to simply guess the number p brings us back to our original problem. This is why we need Quantum computers to solve this problem efficiently.

Quantum computers, unlike their classical counterparts, can simultaneously calculate multiple results for a single input by using Quantum superposition. In Quantum computers the work is based on superpositions that calculate all possible answers at once, while the probabilities of all the wrong answers, represented by states of the qubits interfere destructively and the correct ones interfere constructively, which is exactly what Shor's algorithm does.

Given that Quantum computers pose a problem for asymmetric cryptography, we go back to symmetric cryptography which is secure. But the problem here as discussed earlier was the key distribution through a public channel. To circumvent this problem, we again use the principles of Quantum Mechanics to securely distribute the key through a public channel. Let's see how this is done.

The solution to the problem of key distribution by the introduction of QKD

The main vulnerability of symmetric key cryptography is that an untrusted channel/medium should be used to share the secret symmetric keys. Since the medium is untrusted, it is highly susceptible to attacks by eavesdroppers. From here, the most important question arises, that is, how to securely distribute symmetric keys between two parties?

The solution to this key distribution problem is the quantum key distribution (QKD). It is a cryptographic technique that offers security that is guided and highly guaranteed by the laws of physics. It provides the means to distribute secret keys that can be used with quantum-safe symmetric algorithms like *Advanced encryption standard (AES)* or one-time pad encryption. QKD involves the encoding of information in the quantum states of photons which follows some notions from quantum physics known as key principles of QKD.

Real world quantum key distribution

The performance of a QKD system is described by the rate at which a key is changed over a certain distance — or equally, for a given loss budget. When a photon propagates in an optic fiber, it has a certain probability of being absorbed, despite the high translucency of the glass used. As the distance between two QKD stations increases, the probability of a given photon reaching the receiver decreases. Amiss single- photon sources and sensors further contribute to reducing the number of photons detected by the receiver. still, the fact that only a bit of the photons reach the sensors doesn't constitute a vulnerability, as these don't contribute to the final key. This is only a reduction in the crucial exchange rate

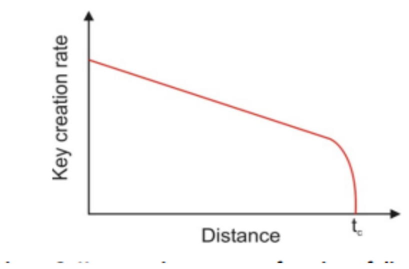


Figure: Key creation rate as a function of distance
(source: google)

As the distance between two stations increases, the two effects reinforce each other to reduce the effective key exchange rate. First, the probability of a given photon reaching the receiver is reduced. This effect causes the raw exchange rate to decrease. Second, the signal-to-noise ratio decreases—the signal decreases with detection probability while the noise probability remains constant—which means the error rate increases. A higher error rate means a more expensive key distillation in terms of the number of bits consumed, and conversely a lower effective key generation rate. However, the key remains safe after distillation

Typical Crucial exchange rates for being QKD systems range from hundreds of kilobits per second over short distances to hundreds of bits per second over longer distances. Because the bits changed by QKD systems are used to induce fairly short encryption keys(128 or 256 bits), the bit exchange rate is sufficient to induce a regular refresh rate of provably secret and absolutely arbitrary keys. The data is also translated using these keys at transmission speeds of over to 10 or indeed 100 Gbps.

The span of current QKD systems is limited by the translucency of optic filaments and generally reaches hundred kilometers(60 long hauls). A much longer distance of 300 km in an optic fiber has been demonstrated. Still, the lower crucial rate attainable for these distances makes real- world operations more grueling . In conventional telecommunications, one deals with this problem by using optic repeaters. They're located roughly every 80 kilometers(50 long hauls) to amplify and regenerate the optic signal. In QKD it isn't possible to do this as optic repeaters would have the same effect as an eavesdropper and lose the key by introducing noise and disquiet. One possible result is to set up a network of trusted bumps, with QKD repeaters to increase the distance. The bumps have to be trusted, and physically secured, because the keys are available at each knot. They can only be set up in secure locales. This is the approach espoused for the Chinese QKD backbone, now installed between Beijing and Shanghai, and in the process of being enforced over a large 11 ’ 000 km long QKD backbone, which will cover the utmost of Eastern China. Some of the links will indeed leave the ground and use optic satellites as trusted bumps, Another approach, which is still in progress, is to replace the trusted bumps with amount repeaters.

Principles used in Quantum Key Distribution (QKD)

Uncertainty principle

In quantum mechanics, the act of measurement is one of the most essential aspects. This uncertainty principle basically tells us that if a measurement is done on a quantum observable of the quantum system, then it will automatically change the other properties of the system. This immediately leads us to the result that it is impossible to measure the values of two non-commuting observables for a quantum system simultaneously.

Let’s say Alice sends the information/quantum states in encoded messages to Bob via a classical communication channel. If an eavesdropper tries to decode the message by doing the quantum measurement, this will change Bob's output state. That means, this principle ensures that no eavesdropper can perform a measurement that keeps the quantum state undisturbed.

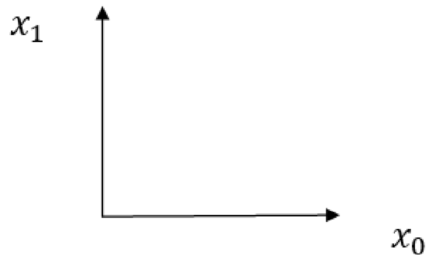
Non-orthogonality principle

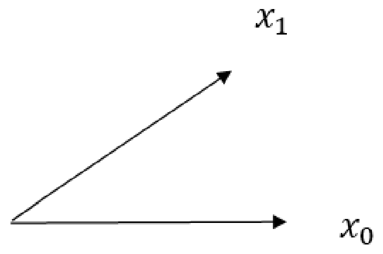
Consider a two-state quantum system, $|\psi_1\rangle$ and $|\psi_2\rangle$. Here

$|\psi_1\rangle = a|0\rangle + b|1\rangle$ where $|a^2| + |b^2| = 1$

$|\psi_2\rangle = c|0\rangle + d|1\rangle$ where $|c^2| + |d^2| = 1$

The non-orthogonality principle says that if $|\psi_1\rangle$ and $|\psi_2\rangle$ are non-orthogonal, that is $\langle\psi_1|\psi_2\rangle \neq 0$, then it is impossible to distinguish between these two states.





Consider the 1st case where x_0 and x_1 encoded in bits 0 and 1 respectively are orthogonal, hence the measurement can distinguish between them easily. Unlike the 1st case, in the 2nd case, the bits 0 and 1 are non-orthogonal. So, there exists no measurement which can distinguish these two states.

This feature is highly useful in Quantum Key Distribution. Because, if the bits 0 and 1 are encoded by non-orthogonal states in the message, then Eve cannot decode the message without making errors.

Authentication Channel

QKD includes the quantum channel to send qubits and a classical channel to figure out the message from the quantum state exchange. But the problem is that the classical channel is a public channel where the message can be heard by the eavesdropper. Hence, the classical channel needs to be authenticated to prevent a man-in-middle attack. In order to solve this issue, an authentication algorithm is used to authenticate Alice and Bob. An authentication algorithm(symmetric in nature) with a pre-shared key is used to authenticate Alice and Bob.

No Cloning theorem

This no-cloning theorem states that it is impossible to perfectly clone an unknown/arbitrary qubit state. That is if we try to clone the state $|\psi\rangle$ onto the state $|0\rangle$ to obtain $|\psi\psi\rangle$, then quantum mechanically it is impossible.

This is one of the important aspects used in QKD. Because this forbids the eavesdropper to make exact copies of the quantum signal. However, the eavesdropper can make approximate or imperfect copies of the quantum state / gain some information about the quantum channel. Hence, Alice and Bob need to erase all the partial information to ensure that they obtain a perfectly-secure key.

How is QKD more beneficial than other key distribution techniques?

Due to the use of principles of quantum mechanics in key distribution, the QKD technique is comparatively more beneficial than other classically available techniques. Firstly, QKD allows the sender and receiver to detect how many errors have been introduced by the eavesdropper and also how much information has been gained by the eavesdropper. Secondly, any kind of attack in QKD has to be done in real-time. That is, no information can be saved for later decryption by the eavesdropper.

BB84 PROTOCOL

BB84 is a QKD Protocol based on the quantum phenomenon of **superposition**. According to Google, **"superposition is the ability of a quantum system to be in multiple states at the same time until it is measured"**. Here Alice will send qubits to Bob through a quantum channel. If Eve tries to evesdrop on the channel, Alice and Bob have a process by which they will be able to detect the presence of Eve in the channel. Hence they will discard the transmitted key and try again to evade Eve.

Steps:

1. Alice will prepare the qubits in either of two bases (+ or ×). The idea is, if the person who measures the qubit, measures it in the right basis (the same basis it was prepared in), he will have a *100% chance of reading its value (either 0 or 1) correctly. If a qubit is measured in the wrong basis, he will get a random result out of 0 and 1.

2. Alice sends these prepared qubits to Bob through a quantum medium.

3. Bob doesn't know which qubit was prepared in which basis. So every time while reading a qubit, he randomly chooses a basis and measures the qubit in that randomly chosen basis. Note that, as there are only 2 bases in which Alice can prepare the qubits, Bob, while reading, will have a 50% chance of reading a qubit in its correct basis.

4. Now, after the Qubits have been read, and destroyed, so that no one can measure them any more, Bob tells Alice the bases in which he measured which qubit. He uses a classical channel to do this. Alice also tells Bob the bases she used to prepare the qubits over the same classical channel. So, both of them know which qubits were measured in the correct base and which weren't. So, now they decide to discard the qubits which were measured in the wrong base and proceed with the ones which are guaranteed to be measured correctly. Note that, the classical medium is a public medium in which everyone can see everyone else's message, but, there is no way they can impersonate one-another.

5. If everything had gone correctly till this point, and Eve hasn't done any of his mischievous activities, Alice and Bob should have two identical keys with them. So, just to verify this fact, Alice randomly chooses some of her bits, and reveals them to Bob through the classical channel. Bob compares them with his own bits.

6. If Eve wasn't there, Bob must have got a *100% match. Else, if Bob measures a reduced match percentage, he can conclude that Eve must have indulged into the process.

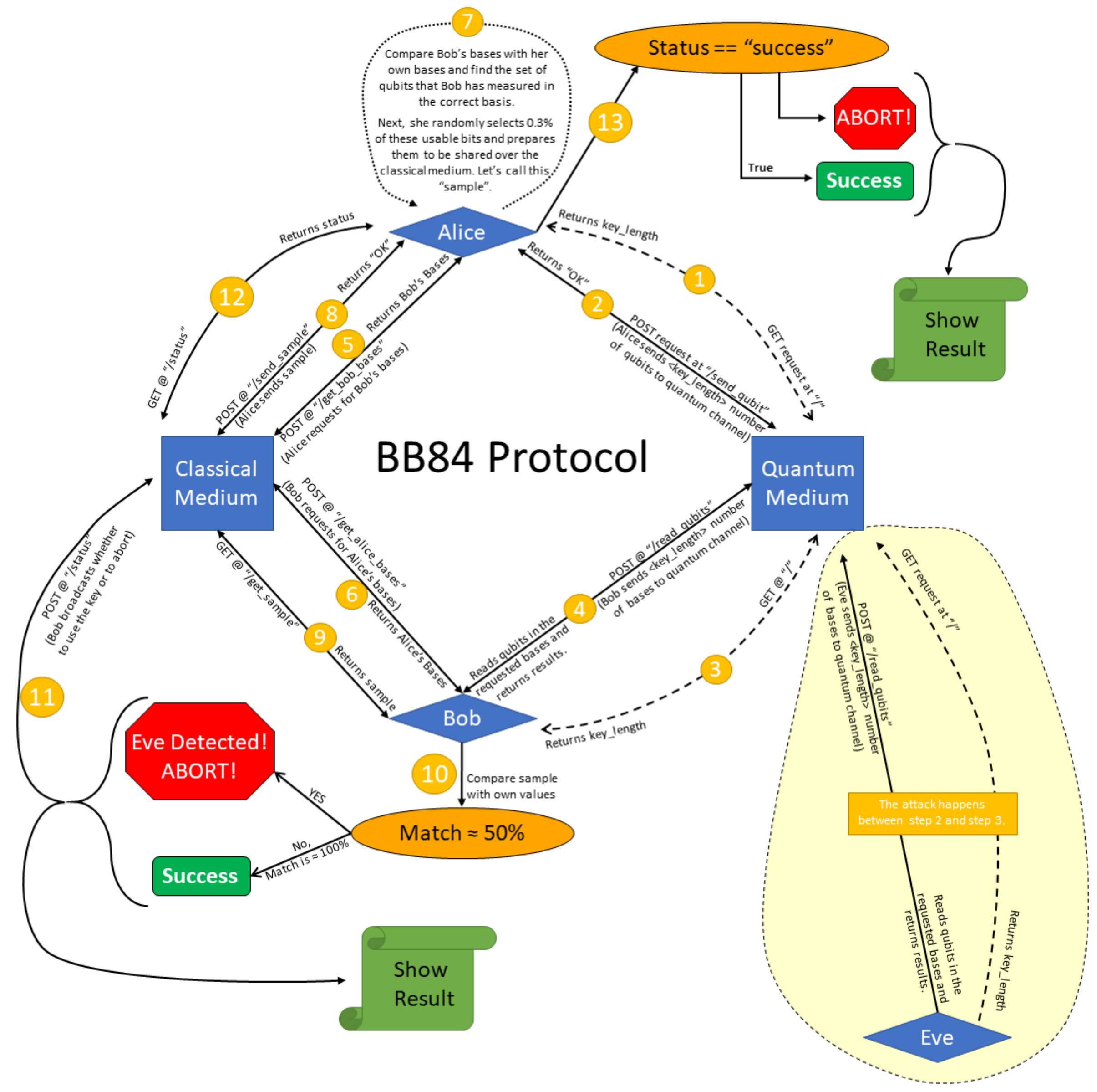
How Eve can try to eavesdrop: intercept and resend

7. Suppose Eve intercepts the quantum medium and measures the qubits in some random choice of basis before Bob. She has a 50% chance of reading the qubits in its correct basis. Now, if Eve doesn't read a qubit in its correct basis, she is bound to disturb its state. The qubit will collapse to the value she reads and on the basis she reads it.

8. Now, let us fast forward to step 5, Bob, compares Alice's sample with his own. Now, among the bases measured in the same basis as Alice had prepared it, Bob will measure the qubits and get the correct value if and only if Eve has also measured them in the correct basis. Chance of that to happen is 50%. So, while doing step 6, if Bob finds that the sample sent by Alice matches his one only by around 50% (which was supposed to be a *100% match), he will declare that the Key was compromised, and the process will again start from step 1.

* **100%:** This value is theoretically 100%, but practically this value goes down by some percent due to errors in quantum systems.

Explanation for the Code:


Compare Bob's bases with her own bases and find the set of qubits that Bob has measured in the correct basis.
Next, she randomly selects 0.3% of these usable bits and prepares them to be shared over the classical medium. Let's call this "sample".
7
Status == "success"
13
ABORT!
True
Success
Show Result
Alice
Returns status
Returns key_length
Returns "OK"
12
8
Returns "OK"
5
Returns Bob's Bases
2
1
GET @ "/status"
POST @ "/send_sample" (Alice sends sample)
POST @ "/get_bob_bases" (Alice requests for Bob's bases)
POST request at "/send_qubit" (Alice sends <key_length> number of qubits to quantum channel)
GET request at "/"
Classical Medium
BB84 Protocol
Quantum Medium
POST @ "/status" (Bob broadcasts whether to use the key or to abort)
GET @ "/get_sample"
POST @ "/get_alice_bases" (Bob requests for Alice's bases)
6
Returns Alice's Bases
9
Returns sample
POST @ "/read_qubits" (Bob sends <key_length> number of bases to quantum channel)
4
Reads qubits in the requested bases and returns results.
GET @ "/"
3
Returns key_length
11
Bob
Eve Detected! ABORT!
10
Compare sample with own values
YES
Match ≈ 50%
No, Match is ≈ 100%
Success
Show Result
POST @ "/read_qubits" (Eve sends <key_length> number of bases to quantum channel)
GET request at "/"
The attack happens between step 2 and step 3.
Reads qubits in the requested bases and returns results.
Returns key_length
Eve

1. Alice asks Quantum Medium to get the length of the Key to be sent.

2. Alice sends that many qubits to the quantum medium.

2.5. (Optional) Eve can ask the Quantum medium for the length of the Key and send those many bases to Quantum Medium to be read.

3. Bob asks Quantum Medium to get the length of the Key to be read.

4. Bob sends those many choices of bases to the quantum medium. The quantum medium reads those bases in the desired medium and returns them to Bob.

5. Alice sends her bases to the Classical Medium, and in return gets Bob's bases

6. Bob also sends his, and in return he gets Alice's bases. (Note, suppose Alice requests the server for Bob's bases by giving her own bases, but Bob hasn't given

his bases to the classical medium yet. So, classical medium will make Alice wait till Bob sends his bases)

7. Make a sample of bases-value pair.

8. Send the Sample to Classical Medium.

9. Bob gets the sample from Classical Medium.

10. Bob matches his own qubits with the sample of Alice.

11. Bob decides whether to Abort or to Use the Key. He tells his decision to the Classical Medium

12. Alice gets to know about the decision made by Bob.

13. Alice acts accordingly.

Simulation of BB84 Protocol using Flask and Qiskit modules of Python. We used these 5 files for this purpose:

1. quantumMedium.py: This is the quantum medium through which the 3 three parties (Alice, Bob and Eve) will communicate with each other.
2. classicalMedium.py: This is the classical medium through which the 3 three parties (alice, bob and eve) will communicate with each other after they have communicated the qubit away.
3. alice.py: This is Alice. She will use this code to carry out the communication throught the two above mentioned mediums.
4. bob.py: Similarly this is the code for Bob.
5. eve.py: Similarly this is the code for Eve.

1. quantumMedium.py

```
from flask import Flask, request
from qiskit import *
from tqdm import tqdm

key_length = 2500
qubits = None
comp = Aer.get_backend("qasm_simulator")

def prepare(inp):
    """Prepares the qubits and returns a list of key_length quantum circuits."""
    bits = inp["bits"]
    bases = inp["bases"]
    message = []
    for i in tqdm(range(key_length)):
        qc = QuantumCircuit(1,1)
```

```python
        if int(bits[i]): qc.x(0)
        if bases[i] == "*": qc.h(0)
        qc.barrier()
        message.append(qc)
    return message

def measure1(qc, basis):
    """Measures the supplied qubit in the desired basis and returns the
result."""
    if basis == "*":
        qc.h(0)
    elif basis != "+":
        raise ValueError('"basis" must be "+" or "*"!')
    qc.measure(0, 0)
    results = list(execute(qc, comp, shots = 1).result().get_counts().keys())[0]
    # if random() > 0.85: results = str(1-int(results))  # adding errors
    return results

def measure(qubits, mbasis):
    """takes all the qubits and all the bases and returns a string containing
measurements."""
    if len(qubits) != len(mbasis):
        raise ValueError(f"length of qubits (= {len(qubits)}) and length of
mbasis (= {len(mbasis)}) aren't equal")
    ret = ""
    for i in tqdm(range(len(mbasis))):
        qc = qubits[i]
        basis = mbasis[i]
        ret += measure1(qc, basis)
    return ret

app = Flask(__name__)

@app.route("/")
def login():
    """Anyone can know the key-length by putting a request to this endpoint."""
    return str(key_length)n
@app.route("/send_qubit", methods=["POST"])
def sendqubit():
    """
    Alice will send a dictionary of the form:
    data = {
            "bits": bits,  # key_length long string of 0s and 1s
            "bases": bases,  # key_length long string of + and *
            "name": name  # here name = "Alice"
            }

    this function will generate quantum circuits from then and store them in a
list
    """
    global qubits
    name = request.form["name"]
    if name == "alice":
        qubits = prepare(request.form)
        return "Qubit Added"
    else:
        return f"{name.title()} is not authorised to send qubits!"

@app.route("/read_qubits", methods=["POST"])
def getqubit():
    """anyone can send a request along with a choice of bases at this
    endpoint to get the qubits read in that base."""
    if not qubits:
        return "Wait"
    bases = request.form["basis"]
    return measure(qubits, bases)

app.run(
    host = "localhost",  # change this to your local ip
      port = 5050,
      debug = True,
 )
```

2. classicalMedium.py

```
from flask import Flask, request

app = Flask(__name__)

# Defining different values
bob_bases, alice_bases = None, None
sample = None
status = None

@app.route('/get_bob_bases', methods=['POST'])
def get_bob_bases():
    """Alice sends her bases and requests for Bob's bases.
    If Bob has already sent his bases, it returns the bases,
    otherwise, it returns "wait", so that Alice waits for
    some time and comes back again to ask for Bob's bases."""
    global alice_bases
    alice_bases = request.form['bases']
    if bob_bases:
        return bob_bases
    return "wait"

@app.route('/get_alice_bases', methods=['POST'])
def get_alice_bases():
    """Similarly Bob sends his bases and requests for Alice's bases.
    If Bob has already sent his bases, it returns the bases, else returns
"wait"."""
    global bob_bases
    bob_bases = request.form['bases']
    if alice_bases:
        return alice_bases
    return "wait"

@app.route("/send_sample", methods=["POST"])
def send_sample():
    """Alice sends her sample."""
    global sample
    sample = request.form["sample"]
    return "Got it!"

@app.route("/get_sample")
def get_sample():
    """Bob requests for Alice's sample
    If Alice has sent her sample, it returns the sample, else returns "wait"."""
    if sample:
        return sample
    return "wait"

@app.route("/status", methods=["GET", "POST"])
def get_status():
    """Bob after knowing Alice's sample, calculates the match factor.
    Depending upon the match factor he either decides to keep the key or not.
    He declares his decision in the classical Medium.
    Later Alice requests for the decision."""
    global status
    if request.method == "GET":
        if status != None:
            return status
        return "wait"
    else:
        status = request.form["status"]
        return "done"

app.run(
    host = "localhost",  # change this to your local ip
       port = 5000,
       debug = True,
 )
```

3. alice.py

```python
from random import choice as IDK
from random import sample as take_any
import requests
from time import sleep

name = "alice"
print(f"Hi! This is {name.title()}!")

qIP, qPORT = "localhost", "5050"
cIP, cPORT = "localhost", "5000"
quantumMedium  =  f"http://{qIP}:{qPORT}/"  # URL of quantum channel
classicalMedium = f"http://{cIP}:{cPORT}/"  # URL of classical channel

def prepare(key_length):
    """prepares key_length number of random choices for qubits
    to be prepared in. a denotes the values of the qubits and b
    denotes the corresponding bases to be used to encode the qubit"""
    a = ""  # X or not
    b = ""  # H or not
    for i in range(key_length):
        a += IDK(["0", "1"])
        b += IDK(["+", "*"])
    return a, b

def sendQubit(key_length):
    """gets the qubit preparation choices and sends them to quantum channel"""
    bits, bases = prepare(key_length)
    requests.post(f"{quantumMedium}send_qubit",
                  data = {
                      "bits": bits,
                      "bases": bases,
                      "name": name
                      }
                  ).text
    return bits, bases

def get_bob_bases(bases):
    try:
        ret = requests.post(f"{classicalMedium}get_bob_bases", data = {"bases":
bases}).text
    except:
        sleep(key_length/120)
        return get_bob_bases(bases)
    if  ret == "wait":
        sleep(key_length/120)
        return get_bob_bases(bases)
    return ret

def send_sample(usable_bits, n = 0.3):
    """prepares and sends a sample of the qubits to quantum channel"""
    sample_indices = sorted(
        take_any(usable_bits, int((key_length/2)*n))
    )
    print("sample length =", len(sample_indices))
    sampleD = {x: bits[x] for x in sample_indices}

    # Encoding it before sending
    sample = ""
    for index, bit in sampleD.items():
        sample += f"{index}:{bit}-"
    sample = sample[:-1]
    # sending
    requests.post(f"{classicalMedium}send_sample", data = {"sample":
sample}).text

    return sampleD

def compile_key(bits, sample, usable_bits):
    """compiles the key from the usable bits and discarding the sample"""
    use = [i for i in range(key_length) if i in usable_bits and i not in
sample.keys()]
    ret = ""
    for i in use:
        ret += bits[i]
    return ret

def successful():
    try:
        ret = requests.get(f"{classicalMedium}status").text
    except:
        sleep(5)
        return successful()
    if  ret == "wait":
        sleep(5)
        return successful()
    if ret == "success":
        return True
    return False



# Know the key length
key_length = int(requests.get(quantumMedium).text)

# Send those many qubit preparation choices to the quantum channel.
# Quantum channel will prepare them as requested
bits, alice_bases = sendQubit(key_length)

# Get the bases of Bob
bob_bases = get_bob_bases(alice_bases)

# know which qubits have been read by Bob in the correct basis
# to calculate which qubits are usable and which aren't
usable_bits = [i for i in range(key_length) if bob_bases[i] == alice_bases[i]]

# Send a sample of the usable qubits to quantum channel
sample = send_sample(usable_bits, n = 0.3)

# If successful, compile the key, else abort the mission
if successful():
    key = compile_key(bits, sample, usable_bits)
    print(f"key = {key}")
else:
    print("Eve Detected, mission abort!")
```

4. bob.py

```
import requests
from random import choice as IDK
from time import sleep

name = "bob"
print(f"Hi! This is {name.title()}!")

qIP, qPORT = "localhost", "5050"
cIP, cPORT = "localhost", "5000"
quantumMedium  =  f"http://{qIP}:{qPORT}/"  # URL of quantum channel
classicalMedium = f"http://{cIP}:{cPORT}/"  # URL of classical channel

def readQubit():
    bases = ""
    for i in range(key_length):
        bases += IDK(["+", "*"])
    # print(basis)
    bits = requests.post(f"{quantumMedium}read_qubits", data = {"basis":
bases}).text
    if bits == "Wait":
        print("waiting...")
        sleep(1)
        return readQubit()
    return [bits, bases]

def get_alice_bases(bases):
    try:
        ret = requests.post(f"{classicalMedium}get_alice_bases", data =
{"bases": bases}).text
    except:
        sleep(1)
        return get_alice_bases(bases)
```

```python
    if  ret == "wait":
        sleep(1)
        return get_alice_bases(bases)
    return ret

def get_sample():
    try:
        ret = requests.get(f"{classicalMedium}get_sample").text
    except:
        sleep(1)
        return get_sample()
    if  ret == "wait":
        sleep(1)
        return get_sample()
    return ret

def process_sample(sample):
    ss = sample.split("-")
    ret = {}
    for s in ss:
        a, b = s.split(":")
        ret[int(a)] = int(b)
    return ret

def compile_key(bits, sample, usable_bits):
    use = [i for i in range(key_length) if i in usable_bits and i not in
sample.keys()]
    ret = ""
    for i in use:
        ret += bits[i]
    return ret

key_length = int(requests.get(quantumMedium).text)

bits, bob_bases = readQubit()

alice_bases = get_alice_bases(bob_bases)


usable_bits = [i for i in range(key_length) if bob_bases[i] == alice_bases[i]]

sample = process_sample(get_sample())

check = [sample[i] == int(bits[i]) for i in sample.keys()]

print(f"match = {sum(check)/len(check)*100}%")

if sum(check)/len(check)>=(0.9-0.05):
    requests.post(f"{classicalMedium}status", data = {"status": "success"})
    print("Success")
    key = compile_key(bits, sample, usable_bits)
    print(f"key length = {len(key)}")
    print(f"key = {key}")
else:
    requests.post(f"{classicalMedium}status", data = {"status": "abort"})
    print("Eve Detected, mission abort!")
```

5. eve.py

```python
import requests
from random import choice as IDK
from time import sleep

name = "eve"
print(f"Hi! This is {name.title()}!")

qIP = "localhost"
cIP = "localhost"
quantumMedium  =  f"http://{qIP}:5050/"  # quantum channel in port 5050
classicalMedium = f"http://{cIP}:5000/"  # classical channel in port 5000

def readQubit():
    bases = ""
    for i in range(key_length):
        bases += IDK(["+", "*"])
    # print(basis)
    bits = requests.post(f"{quantumMedium}read_qubits", data = {"basis":
bases}).text
    if bits == "Wait":
        print("waiting...")
        sleep(1)
        return readQubit()
    return [bits, bases]

key_length = int(requests.get(quantumMedium).text)

bits, bob_bases = readQubit()

print("completed eavesdropping!")
```

On the EPR paradox and the motivation for Bell's inequality

Einstein, Prodolsky and Rosen argued that Quantum mechanics could not be a complete theory and that it should be supplemented by additional variables. Their argument is as follows:

Consider a spin zero meson decaying into an electron and a positron. These two particles move in the opposite direction. Now owing to the conservation of Angular Momentum, which is sacrosanctum, it requires that the total spin of the combined electron-positron system is zero. This forms what many physicists believed known as a singlet entangled state which EPR were not happy about.

The measurements can be made on selected components of the spin of individual particles. Alice makes the measurement on the electron and Bob makes the measurement on the positron. We denote the $\vec{\boldsymbol{\sigma}}.\hat{a}$ as the spin measurement operator on the electron in the $\hat{a}$ direction and $\vec{\boldsymbol{\tau}}.\hat{b}$ as the spin measurement operator on the positron in the $\hat{b}$ direction. Notice that we use the symbols sigma and tau for the electron and positron respectively to keep the notation simple. Let MA be the measurement outcome by Alice and MB be the measurement outcome by Bob, the measurement outcome is always ±1. If Alice performs the measurement $\vec{\boldsymbol{\sigma}}.\hat{a}$ and the measurement outcome M_A is +1, then it follows from angular momentum principle that the measurement $\vec{\boldsymbol{\tau}}.\hat{a}$ by Bob is -1 and vice-versa. The crucial assumption that EPR make here is the locality assumption, by this they mean that, if the two measurements by Alice and Bob are made remotely, the outcome of Alice's measurement doesn't influence the outcome of Bob's.

Again from the conservation of Angular momentum, we could predict in advance the result of measurement of any chosen component of $\vec{\boldsymbol{\tau}}$ (positron's spin), by formerly measuring the same component of $\vec{\boldsymbol{\sigma}}$ (electron's spin). The word any is the key here. Since we can predict in advance the result of measurement of ANY chosen component of the positron's spin, it should follow that the result of these measurements were predetermined, meaning that both particles had well-defined spins in all the directions from the time they were created. This is the element of reality of each particle.

The idea that the electron and positron had well-defined spins all along from the time they were created is the realist's view which EPR subscribed to. But the orthodox view holds that neither particle had a well-defined spin until the act of measurement forced it to take a particular spin direction. It is in an entangled state.

The EPR's argument is that if the spins were not predetermined as the orthodox view claims and we made the measurements on electron and positron simultaneously when they are 100-lightyears away, it would imply that upon measuring the electron's spin along the a direction, the result of the measurement was reaching the positron instantaneously thus producing the positron's spin measurement result in accordance with the conservation of angular momentum leading to the violation of the locality principle.

This instantaneous communication is what Einstein called preposterous. Now, owing to the postulate that no influence can propagate faster than the speed of light, let's just stick to the realist's view and say that both the particles had well-defined spins all along. Now, it follows that If the particles had predetermined spins all along from the time of their creation and Quantum mechanics as a theory could not completely describe the state of either particle in the sense that all it talked about was superposition and probabilities and didn't definitely predict the outcomes of any measurement, it implies that there's a possibility of more complete specification of the state. Their argument, therefore was that the Quantum theory is an incomplete theory, the wavefunction is an incomplete description of the reality, there's some quantity **λ** in addition to Ψ, that is required to specify the state of the system completely.

λ is the local hidden variable that represents the elements of reality associated with the spins. So, for each electron-positron system created, there's a particular **λ** that is fixed and local to each particle and it contains in itself the information to yield the results of spin measurements in all the direction from the time the particles were created. Bell's genius lies in the fact that he came up with a relation that proves that no local hidden variable theory can explain the results of the experiments. We will look at what exactly the experiments reveal.

Let's look a little more into why the orthodox physicists believed the description of the reality was as complete as an entangled state is.

Charlie performs the pion decay experiment and her role now is to distribute the electron and positron to Alice and Bob respectively. The wave function governing this composite system, according to the orthodox physicists is called the entangled singlet state given by:

$$|\Psi sing > \ = \ 1/\sqrt{2}\,(\,|\uparrow\downarrow > \ - \ |\downarrow\uparrow >)$$

To them, this wave function is as complete a description of the combined system as it is possible to make.

These electron-positron pairs are generated several times, Alice performs the measurement in the z direction, she obtains +1 in 50 percent of the trials and -1 in the other 50 percent of the trials, the same goes with bob. We have observed that we can know nothing at all about the outcome of the individual measurement of any component of the spin when a pion decays into electron-positron pair.

How does mathematics express this fact using the entangled wave function? To see this, we calculate the expectation of the $\vec{\boldsymbol{\sigma}}.\hat{z}$

$$\langle M_A.M_B \rangle \quad = \quad <\Psi sing|\vec{\boldsymbol{\sigma}}.\hat{z} \otimes I\,|\Psi sing>$$

$$= \quad 1/2\ (<\uparrow\downarrow| \ - \ <\downarrow\uparrow|)\,|\vec{\boldsymbol{\sigma}}.\hat{z} \otimes I|\,(\,|\uparrow\downarrow> \ - \ |\downarrow\uparrow>)$$

$$= \quad 1/2 \ + 0 \ + 0 \ + \ 1/2$$

$$= \quad 0$$

This can be shown for $\vec{\boldsymbol{\sigma}}.\hat{z}$, $\vec{\boldsymbol{\sigma}}.\hat{y}$,$\vec{\boldsymbol{\sigma}}.\hat{x}$ or $\vec{\boldsymbol{\tau}}.\hat{z}$, $\vec{\boldsymbol{\tau}}.\hat{y}$,$\vec{\boldsymbol{\tau}}.\hat{x}$ or in general for any direction. The expectation value zero implies that the experimental outcome is equally likely to be +1 or -1, this very well agrees with the experiments that the outcome of an individual measurement is completely uncertain. So, we can know everything about this composite system - everything there is to know-- that's the complete description of the wave function and we can still know nothing about its constituent parts, this is weirdness of this electron-positron pair generated from a pion. This is what they called entanglement.

Knowing the exact state of the system, even if it is entangled, must tell us something. And in fact it does, when we consider the composite observables. Now let's say both alice and bob measure in the z direction simultaneously, the observable therefore is given by $\vec{\boldsymbol{\sigma}}.\hat{z} \ \otimes \ \vec{\boldsymbol{\tau}}.\hat{z}$, the expectation is now given by

$$\langle M_A.M_B \rangle \quad = \quad <\Psi sing|\vec{\boldsymbol{\sigma}}.\hat{z} \ \otimes \ \vec{\boldsymbol{\tau}}.\hat{z}\ |\Psi sing>$$

$$= \quad 1/2\ (<\uparrow\downarrow| \ -<\downarrow\uparrow|)\,|\vec{\boldsymbol{\sigma}}.\hat{z} \otimes \vec{\boldsymbol{\tau}}.\hat{z}\ |\,(\,|\uparrow\downarrow> - \ |\downarrow\uparrow>)$$

$$= \quad 1/2\ (<\uparrow\downarrow| \ -<\downarrow\uparrow)|\,(- \ |\uparrow\downarrow> + \ |\downarrow\uparrow>)$$

$$= \quad -1$$

The expectation of the product of the measurement outcomes of Alice and Bob is called the correlation. Experimentally,it means that when both of them measure the spin in the same direction z, they come back, compare their recorded outcomes in all the trials, and see that they have measured opposite values and therefore the product is always -1. Sometimes Alice measures +1 and Bob measures -1, other times viceversa. Taking the expectation over several trails indeed gives -1 experimentally as well. This result is also correctly predicted by the entangled wave function as shown in the above calculation.

In fact, this is nothing surprising, Einstein would argue that this could be a result of a completely classical setup where the particles have a physical reality. It's as if

Charlie prepared two particles, one in spin up and the other spin down and distributed them to Alice and Bob in each trail without knowing which is which. Here as well, sometimes Alice gets +1 and Bob gets -1, other times vice-versa and therefore leads to the same correlation.

But then we come to something that has no classical analogy, instead of measuring

$\vec{\boldsymbol{\sigma}}.\hat{z} \otimes \vec{\boldsymbol{\tau}}.\hat{z}$, alice and bob measure in the x basis, $\vec{\boldsymbol{\sigma}}.\hat{x} \otimes \vec{\boldsymbol{\tau}}.\hat{x}$,

$$
\begin{aligned}
\langle M_A.M_B \rangle &= \langle \Psi sing|\, \vec{\boldsymbol{\sigma}}.\hat{x} \otimes \vec{\boldsymbol{\tau}}.\hat{x}\, |\Psi sing \rangle \\
&= 1/2\ (\langle \uparrow\downarrow| - \langle \downarrow\uparrow|)|\vec{\boldsymbol{\sigma}}.\hat{x} \otimes \vec{\boldsymbol{\tau}}.\hat{x}\,|\,(|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle) \\
&= 1/2\ (\langle \uparrow\downarrow| - \langle \downarrow\uparrow)|\,(|\downarrow\uparrow\rangle - |\uparrow\downarrow\rangle) \\
&= -1
\end{aligned}
$$

This is such a startling result, but nonetheless agrees with the experiments. Startling because in a classical setup, if a spin is measured in the z direction and it gives you +1 or -1, it means that the spin of the particle is aligned in the z direction and if we measured again in the x basis, a classical setup would straight out throw at us a zero for both Alice and Bob and the correlation would be zero. But what we measure experimentally is -1. This is the absolute weirdness of these particles, the orthodox opinion is not just another opinion, it is an informed opinion. At Least as long as we have not found any hidden variable that represents the element of physical reality which reproduces the quantum mechanical correlation and still satisfies the principle of locality.

In general it can easily be derived from the entangled state using the mathematics of quantum mechanics that when alice and bob choose the directions arbitrarily, the correlation $\langle M_A.M_B \rangle = -\hat{a}.\hat{b} = -\cos(\theta)$ where θ is the angle between $\hat{a}$ and $\hat{b}$ direction which very well agrees with the experimental results.

Sagnik, add it here

We know that when both of them measure in the same basis, they get opposite results, this can be mathematically

This correlation has been experimentally verified. To arrive at this correlation theoretically, we indeed used the entangled state. The entangled state, as argued by the EPR violates locality. So the question John Bell asked is, is it possible to arrive at this correlation with a local hidden variable model? Bell eventually answered the question; he came up with a theorem that proves any local hidden variable theory cannot reproduce this correlation. Therefore the entangled state is the complete description of reality.

CHSH inequality

Consider the relation:

$$C = (MA + MA')MB + (MA - MA')MB'$$

Where MA and MA' are the results of measurement by Alice in the A and A' direction respectively, MB and MB' are the results of measurement by Bob in the B and B' direction respectively. These results can take values +1 or -1.

C can be rewritten as

$$C = MA.MB + MA'.MB + MA.MB' - MA'.MB'$$

Taking the average

$$\langle C\rangle = \langle MA.MB\rangle + \langle MA'.MB\rangle + \langle MA.MB'\rangle - \langle MA'.MB'\rangle$$

The way we take the average is, for eg: Charlie conducts a million experiments thereby generating a million electron-positron pairs. These are distributed to Alice and Bob one by one for measurement. They randomly pick the measurement direction in each trial. After each trial, they note down their measurement direction and the result in their respective notebooks. After a million trials, they both come back, look at each other's notebook to calculate the above 4 quantities. For eg: To calculate <MA.MB>, they count the number of trials where Alice picked A and Bob picked B, and then calculate the value MA.MB of all these trials, remember that MA and MB can take values from {+1,-1}, therefore MA.MB can take +1 or -1. Finally they add the MA.MB value of all these trials and divide by the number of trials. We will try to upper bound <C> in a local hidden variable , according to it, the results of measurements in all these directions are predetermined for every electron-positron pair. Therefore these are the possible combinations of measurement results

MA	MA'	MB	MB'
-1	-1	-1	-1
-1	-1	-1	1
-1	-1	1	-1
-1	-1	1	1
-1	1	-1	-1

-1	1	-1	1
-1	1	1	-1
-1	1	1	1
1	-1	-1	-1
1	-1	-1	1
1	-1	1	-1
1	-1	1	1
1	1	-1	-1
1	1	-1	1
1	1	1	-1
1	1	1	1

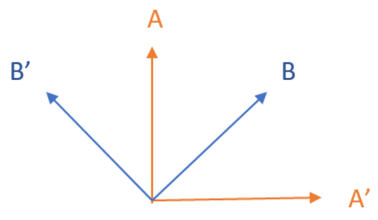
Every electron pair released has one of the above configurations. When MA = MA', MA + MA' = ±2, this implies |C| = |(MA + MA')MB + (MA - MA')MB'| = 2. Similarly when MA = -MA', MA + MA' = 0, this also implies |C| = |(MA + MA')MB + (MA - MA')MB'| = 2.

Therefore, in any local hidden variable theory, we can bound |<C>| as following

$$|\langle C\rangle| \leq \langle |C| \rangle = 2$$

$$|\langle C\rangle| = |\langle MA.MB\rangle + \langle MA'.MB\rangle + \langle MA.MB'\rangle - \langle MA'.MB'\rangle| \leq 2$$

The Above relation is called the CHSH inequality. Now consider the directions as seen in the picture. When these directions are chosen by Alice and Bob in the experiments. They see that <MA.MB> = -1/√2 = - A.B which is exactly predicted by the mathematics of Quantum mechanics involving the entangled wave function. Similarly, <MA'.MB> = -1/√2 , <MA.MB'> = -1/√2, <MA'.MB'> = 1/√2. Calculation of |<C>| yields a value of 2√2 Which is a clear violation of the bounds predicted by the local hidden variable theory. Therefore, Bell concluded that no local hidden variable theory can reproduce the correlations predicted by quantum mechanics which is in conjunction with the experiments in all the angles.



Now, we are compelled to agree that the wave function governing the electron-positron pair is indeed an entangled wave function, i.e; neither particle has a well defined spin. It is indeed as though the result of measurement of one particle is instantaneously communicated to the other. This was the shock, the nature itself is fundamentally nonlocal.

We can use entangled particles for key distribution, that is the basis of E91 protocol. We can use the chsh inequality to verify the entanglement

E91 PROTOCOL

1) Say, Alice wants to communicate with Bob. They assign the job of generating the entangled particles in a singlet state to Charlie

$$|\Psi sing > \;=\; 1/\sqrt{2}\,(\,|\uparrow\downarrow > \;-\; |\downarrow\uparrow >)$$

2) Charlie generates and distributes them to Alice and Bob. When they both measure in the same basis, the probability that Alice measures +1 and Bob measures -1 is ½ , similarly the probability that Alice measures -1 and Bob measures +1 is also ½ . The probability that both of them measure the same is always zero.

3) They assign bit 0 to +1 result and bit 1 to -1 result. As long as they are measuring in the same basis, their results are exactly opposite. They use this fact to establish the key.

Now that we have understood this, the protocol is as follows.

1) After every entangled pair generated by Charlie, Alice randomly chooses from the directions on the left circle and Bob from the right circle for their respective measurements.

2) Upon each measurement, both of them note down the basis they have used and tabulate them as shown below as an example for 10 measurements.

Alice's basis	A1	A2	A3	A2	A1	A3	A2	A1	A2	A3
Bob's basis	B1	B3	B3	B1	B2	B2	B3	B2	B1	B3
	✔		✔							✔

3)After all the measurements are made, they exchange information about the basis they used.

4)Both of them retain the results where they used the same basis for the key generation, indicated by the check marks above. In all these trials, it is assured that they have the opposite results. So Bob simply inverts his result, which then matches exactly with Alice's, now he assigns 0s and 1s, this way both of them have the same bits and thus they have generated the key.

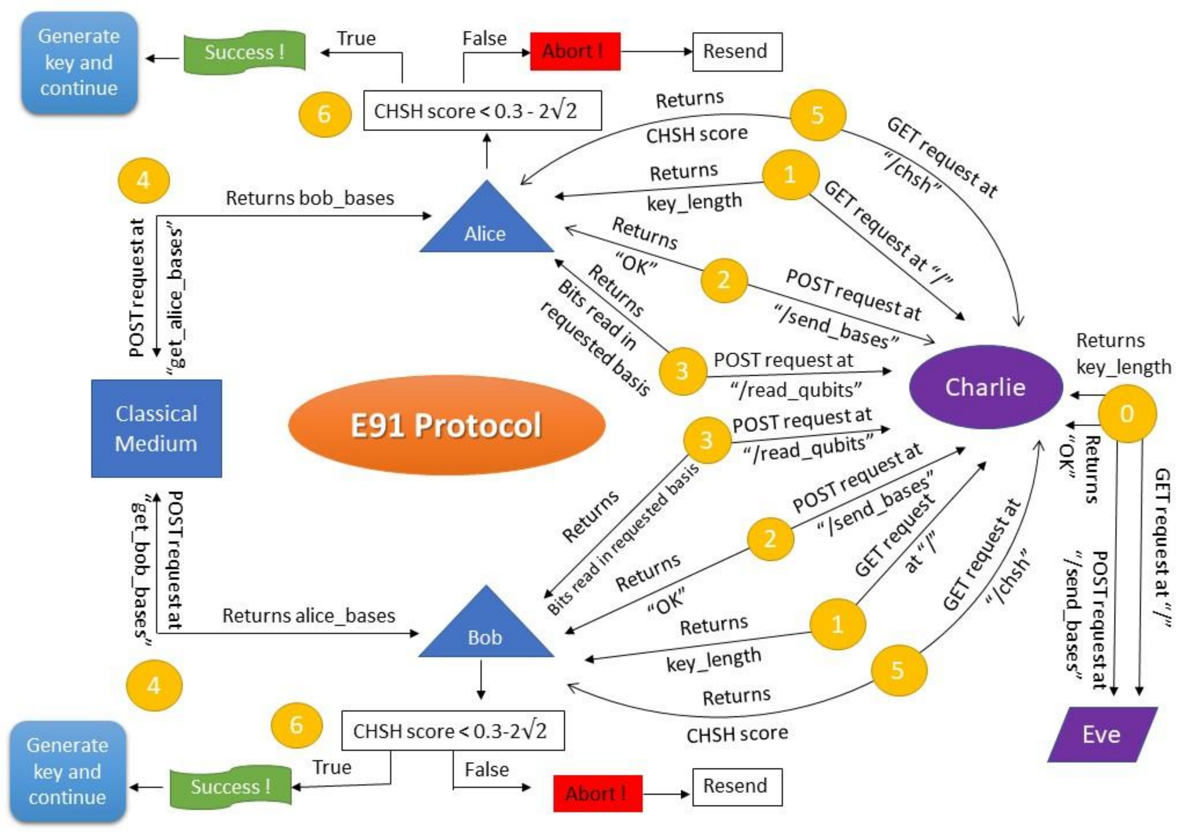
5) But how do they know they have the same key? Remember that by using the above process they can generate the same key only when the particles are entangled. To test whether the particles are entangled they use the remaining measurement results

6) They calculate <MA1.MB3>,

These are ht

Note that there are three measurement directions to choose from for both Alice and Bob unlike in BB84, the purpose of this shall be explained

Both of them agree beforehand which measurement directions they use to establish the key.



Code for E91:

We used these 6 files for this purpose:

6. quantumMedium.py: This is the quantum medium through which the 3 three parties (Alice, Bob and Eve) will communicate with each other.
7. classicalMedium.py: This is the classical medium through which the 3 three parties (alice, bob and eve) will communicate with each other after they have communicated the qubit away.
8. alice.py: This is Alice. She will use this code to carry out the communication throught the two above mentioned mediums.
9. bob.py: Similarly this is the code for Bob.
10. eve.py: Similarly this is the code for Eve.

Appendix:

1. All the Codes have been kept in my GitHub Repository: https://github.com/PeithonKing/Attacking_QKD_Protocols
2. Asymmetric and symmetric cryptosystems: Quantum cryptography (Nicolas Gisin, Gregorie Ribordy, Wolfgang Tittle and Hugo Zbinden)