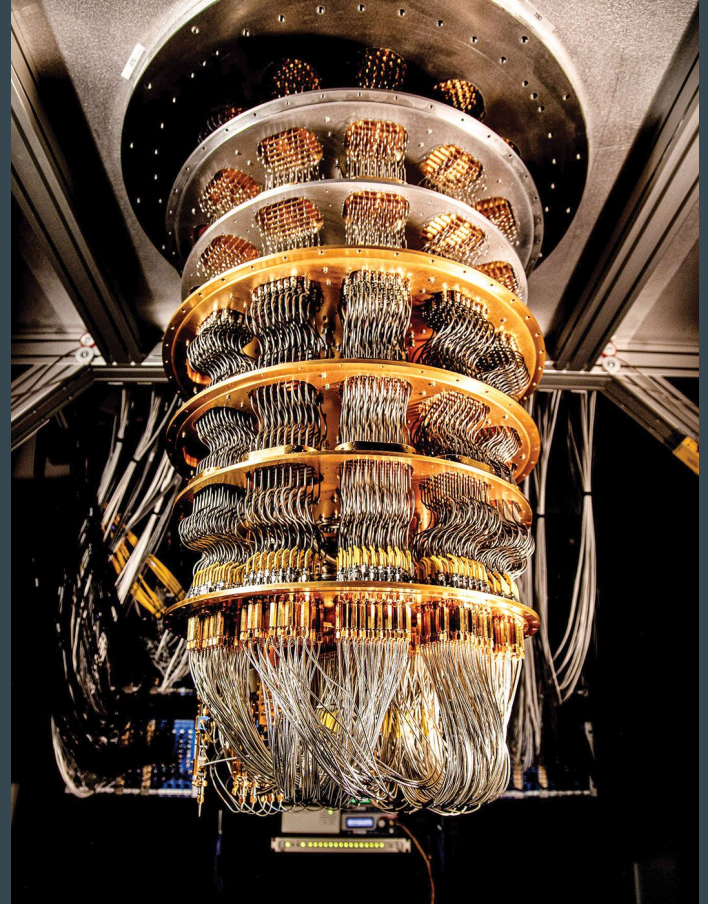


# DATA PROTECTIONS

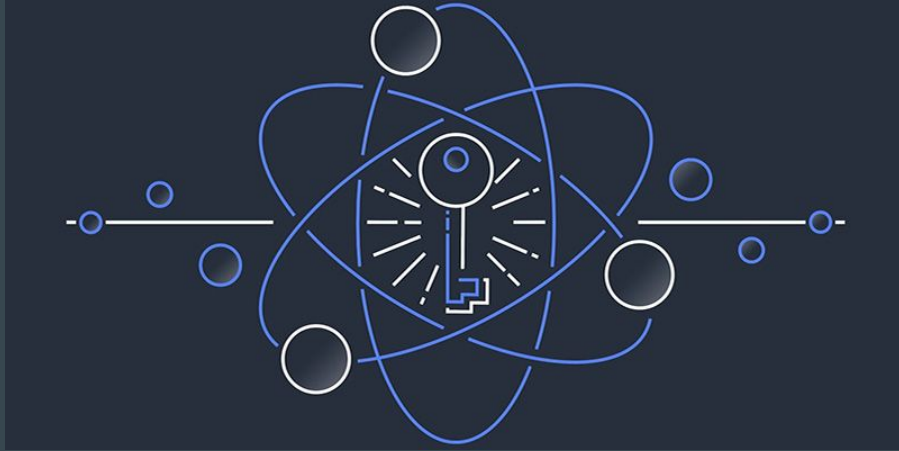
TEAM QUANTUM MANIAC ...

TEAM MEMBERS:-

- NAMAN KAUSHIK
- MD SAJIDULLAH ANSARI



# TIMELINE:-



**DES ENCRYPTION**



**AES ENCRYPTION**



**Quantum  
cryptography**

# PROBLEM STATEMENT

As a Banking Firm, You always need to be data proof because these are sensitive data to deal with and with the advent of quantum computing, traditional encryption methods are becoming increasingly vulnerable to attacks. Therefore, there is a growing need for developing quantum-safe encryption algorithms to protect sensitive data from potential breaches. The challenge is to find efficient and reliable methods of implementing quantum-safe encryption techniques that can secure the transfer and storage of sensitive data. The objective is to explore the potential of quantum cryptography and related technologies to provide a secure platform for data protection in the face of quantum computing threats.

# SOLUTION

QKD can provide unconditional security against any computational attack, including quantum attacks, making it a promising solution for secure communication. QKD is a method of generating and sharing secret keys between two parties based on the principles of quantum mechanics. Unlike classical key exchange protocols, QKD uses the fundamental laws of physics to protect data transmission against eavesdropping.

# METHODOLOGY

**ENCRYPTION**



**Quantum key  
distribution**



**DECRYPTION**

# STEPS:-

1. We have implemented a secure system for encrypting and decrypting data.
2. The Advanced Encryption Standard (AES) provides strong encryption for sensitive data using a secret key.
3. The BB84 protocol is used for secure key transfer by leveraging the principles of quantum mechanics.
4. By using both AES and BB84, you have created a secure system for transmitting and decrypting data.
5. BB84 is used to detect any attacks or interference during the key transfer process, adding an additional layer of security to the system.

## AES algorithm:-

- AES is a symmetric-key encryption algorithm that is widely used for securing data.
- It operates on fixed-length blocks of data and uses a key to transform plaintext into ciphertext and back.
- The key length can be 128, 192, or 256 bits, and the block length is fixed at 128 bits.
- The AES algorithm consists of several rounds of operations, including byte substitution, shift rows, mix columns, and add round key.
- AES is widely accepted and trusted for its strong security properties and efficiency in terms of computation time and memory usage.

WORKING  
PROTOTYPE



# ENCRYPTED DATA

The screenshot shows a Jupyter Notebook interface with a dark theme. At the top, there's a menu bar with 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', and a dropdown arrow. Below the menu bar, there's a toolbar with icons for file operations and a search icon. The main area is divided into two sections. The top section contains a code cell with the following Python code:

```
file2=open(sys.argv[2], "r")
print("The output hex value for the entire message is:\n%s\n" % file2.read())
file2.close()
```

The output of this cell is a long hexadecimal string: 736b0e8aecc15cbea9f6420b02203545d590ef16ebbf6f0896493333c56ec8eb86b28ce490a2bf0e10906a262ce2e460750b33ba878470f281cf56f192ec86b4ba1742b68cba956f73497c531595127d7:

The bottom section contains a code cell with the following Python code:

```
alice_basis = np.random.randint(2, size = n_qubits)
bob_basis = np.random.randint(2, size = n_qubits)

print(alice_random_string)
print(alice_basis)
print(bob_basis)
```

The output of this cell is three lines of text: alice\_random\_string, alice\_basis, and bob\_basis.

At the bottom of the interface, there's a status bar with 'master' and 'tabnine starter' labels, and a 'Cell 2 of 10' indicator.

# RECEIVER ENTERING THE KEY

```
File Edit Selection View Go Run ...
BB84.ipynb U plaintext1.txt M Ur Enter the key (Press 'Enter' to confirm or 'Escape' to cancel)
Untitled7.ipynb > # PassPhrase=[]
+ Code + Markdown | Interrupt Clear All Outputs Go To Restart | Variables Outline ... Python 3.9.13

    possible_key = possible_key + str(bob_key[index])
else:
    possible_key = possible_key + '-'
print(possible_key)

keys.append(possible_key)

[130] ✓ 0.0s

... --1000--
--1000--
--1000--
--1000--
--1000--
--1000--
--1000--
--1000--
--1000--

info=keys[0]
arr_info=[]
for i in info:
    arr_info.append(i)
ans=""
arr=[]
input_key=input("Enter the key")
for i in input_key:
    arr.append(i)

for i in range(0,len(info)):
    if arr_info[i] == "-":
        ans=ans+arr[i]
    else:
```

Cell 2 of 10 Go Live

# DECRYPTED DATA

The screenshot shows a Jupyter Notebook environment with the following components:

- File Explorer:** Contains files `BB84.ipynb`, `plaintext1.txt`, `Untitled7.ipynb`, `plaintext2.txt`, `BitVector.py`, and `AESdecrypt.py`.
- Code Editor:** Displays the following Python code in `Untitled7.ipynb`:

```
# PassPhrase=[]

outputhex = BitVector(hexstring=myhexstring)
asciioutput = outputhex.get_bitvector_in_ascii()
asciioutput=asciioutput.replace('\x00','')
FILEOUT.write(asciioutput)

FILEOUT.close()

file2=io.open(sys.argv[2], "r", encoding='utf-8')
print("The decrypted message for the entire ciphertext is:\n%s\n" % file2.read())
file2.close()
```
- Output:** The execution of the code produced the following output:

```
[132] ✓ 19.0s

... Enter in the 16 character passphrase to decrypt your text file ciphertext.txt
Inside your ciphertext message is:
736b0e8aecc15cbea9f6420b02203545d590ef16ebbf6c6f0896493333c56ec8eb86b28ce490a2bf0e10906a262ce2e460750b33ba878470f281cf56f192ec86b4ba1742b68cba956f73497c531595127d7:

The decrypted message for the entire ciphertext is:
name  pokedex id  height  weight  type  secondary type  hp    attack  defense  sp  atk  sp def  speed
bulbasaur      1      7      69      grass  poison  45      49      49      65      65      45
ivysaur 2     10     130     grass  poison  60      62      80      80      60      80
venusaur      3      20     1000    grass  poison  80      82      83     100     100     80
charmander     4      6      85      fire   None    39      52      43      60      50      65
charmeleon     5     11     190     fire   None    58      64      58      80      65      80
charizard      6     17     905     fire   flying  78      84      78     109     85     100
squirtle       7      5      90      water  None    44      48      65      50      64      43
wartortle      8     10     225     water  None    59      63      80      65      80      58
blastoise     9     16     855     water  None    79      83     100     85     105     78
```
- Bottom Bar:** Shows the status bar with "master", "0 0 0", "tabnine starter", "Cell 2 of 10", "Go Live", and other icons.

# INTERCEPTION DETECTION

```
# PassPhrase=[]
n_qubits = 8
indx=0
PassPhrase=""
j=""
alice_random_string = np.random.randint(2, size = n_qubits)
for i in alice_random_string:
    j=str(i)
    PassPhrase = PassPhrase + j
print(PassPhrase)
```

[172] ✓ 0.0s

... 00110010

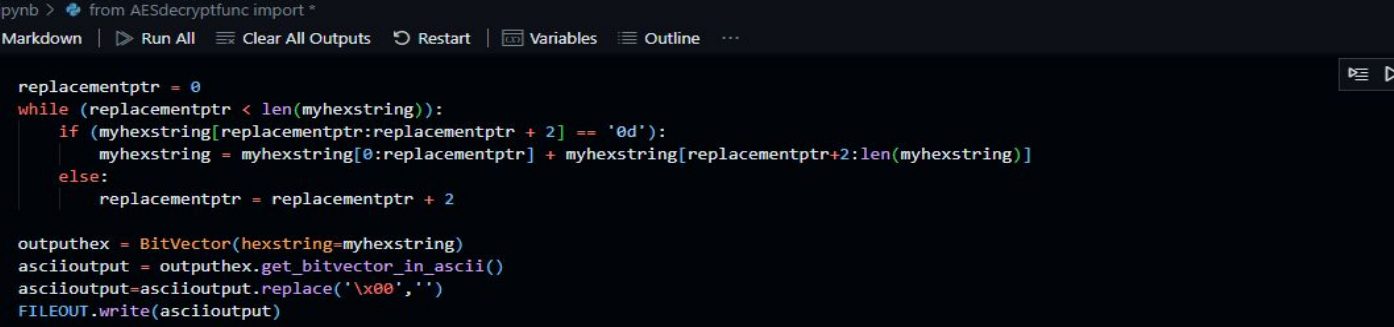
```
File Edit Selection View Go Run ...
BB84.ipynb U plaintext1.txt M 00110010 (Press 'Enter' to confirm or 'Escape' to cancel)
Untitled7.ipynb > # info=keys[0]
+ Code + Markdown | Interrupt Clear All Outputs Go To Restart Variables Outline ...
else:
    possible_key = possible_key + '-'
    print(possible_key)
    keys.append(possible_key)

[178] ✓ 0.0s
... 00--0--0
    00--0--0
    00--0--0
    00--0--0
    00--0--0
    00--0--0
    00--0--0

# info=keys[0]
info="00--0-00"
arr_info.append(i)
ans=""
arr=[]
input_key=input("Enter the key")
for i in input_key:
    arr.append(i)

for i in range(0,len(info)):
    if arr_info[i] == "-":
        ans=ans+arr[i]
    else:
        if arr_info[i] == arr[i]:
            ans = ans+arr[i]
```

# DECRYPTED DATA AFTER INTERCEPTION



The screenshot shows a Python IDE with a dark theme. The top bar includes a search icon and the text 'AES-Encryption-Python'. The file explorer on the left shows a project named 'AES-Encryption-Python' with files 'BB84.ipynb', 'plaintext1.txt', 'plaintext2.txt', 'BitVector.py', and 'AESdecrypt.py'. The main editor displays the code for 'AESdecrypt.py', which imports 'from AESdecryptfunc import \*' and contains a function 'decrypt' that takes a ciphertext file path as input. The function reads the ciphertext, iterates through it, and decrypts it using a passphrase. The output is written to a file named 'plaintext2.txt'. The console shows the execution of the script, which prompts for a passphrase and displays the decrypted message.

```
replacementptr = 0
while (replacementptr < len(myhexstring)):
    if (myhexstring[replacementptr:replacementptr + 2] == '0d'):
        myhexstring = myhexstring[0:replacementptr] + myhexstring[replacementptr+2:len(myhexstring)]
    else:
        replacementptr = replacementptr + 2

outpuhex = BitVector(hexstring=myhexstring)
asciioutput = outpuhex.get_bitvector_in_ascii()
asciioutput=asciioutput.replace('\x00','')
FILEOUT.write(asciioutput)

FILEOUT.close()

file2=io.open(sys.argv[2], "r", encoding='utf-8')
print("The decrypted message for the entire ciphertext is:\n%s\n" % file2.read())
file2.close()
```

[180] ✓ 23.5s Python

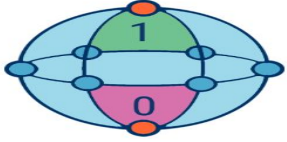
... Enter in the 16 character passphrase to decrypt your text file ciphertext.txt  
Inside your ciphertext message is:  
d8214864ca048197b341005f590d3ef999028480106d06f68d62f3187c7af2252dea560d645dc758156bb7a89666da9276ab4aeb76ca33b8aa51fcfd05fa1745cc006c936c268b678370cb2a32cc08d5a7f

The decrypted message for the entire ciphertext is:  
00207e0000b80'E:0160YÊaqXÊ0Ëy00'0'U=‰0-·?KkX00X0<\$0%0N0MP00x0p0é000'0Ny0N úú0000É=Ï;50Ï00000<000'D0UP000·:0\(\uÍ>0000000\_0iA »0V300p0tú00V0ý000KBA0~u0A40000>8000Áh'  
0000"cg.~/|90z00n#k000ÁÍ0ú°<>0ZT002800\$00000"q00Ï00úq9°670 00n06±0U0K4Mìt=.U)00T±000000á>;ñúj000'ÏQ0·P00vL00Dx ÊÊ»000ñy00j00Û000<0) y:×.0YR00Ï0Ïu00»(É00V%4t

# Quantum Computing

Vs.

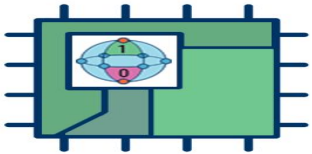
# Classical Computing



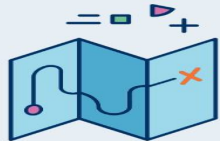
Calculates with qubits, which can represent 0 and 1 at the same time



Power increases exponentially in proportion to the number of qubits

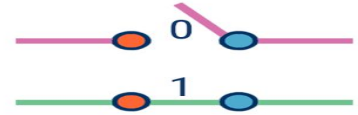


Quantum computers have high error rates and need to be kept ultracold



Well suited for tasks like optimization problems, data analysis, and simulations

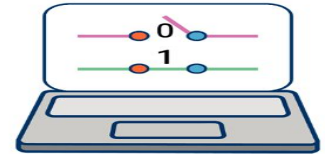
Calculates with transistors, which can represent either 0 or 1



Power increases in a 1:1 relationship with the number of transistors



Classical computers have low error rates and can operate at room temp



Most everyday processing is best handled by classical computers





# advantages of quantum computing :-

- **Faster computation:** Quantum computer excels in some specialized tasks over a supercomputer. A currently existing quantum computer developed by Google is 158 million times more powerful than a supercomputer.
- **Lower power consumption:** The power consumption of a quantum computer is less than that of a supercomputer due to quantum tunneling.
- **Simulation:** Quantum computers are more efficient at modeling natural systems at the quantum level and can simulate the dynamics of quantum systems more accurately.

# QUANTUM KEY DISTRIBUTION

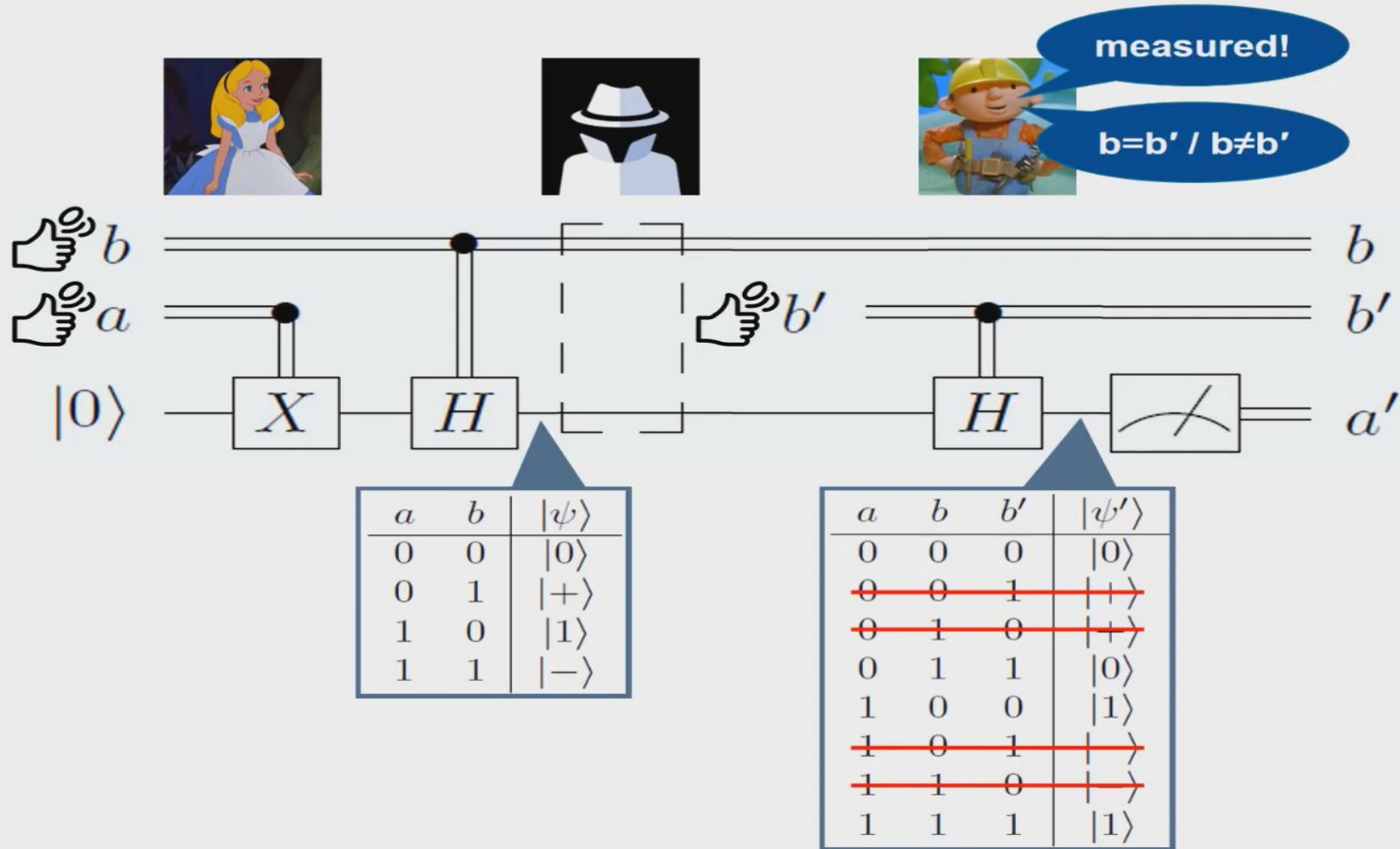


# BB84 ALGORITHM:-

The methodology for implementing the BB84 protocol involves the following steps:

1. Create a random binary key.
2. Encode the key into quantum states using randomly selected bases.
3. Transmit the quantum states over a secure communication channel.
4. Measure the received quantum states using randomly selected bases.
5. Compare a subset of the key bits to detect any errors or eavesdropping attempts.
6. Use privacy amplification techniques to distill a smaller, but more secure key.

# BB84 PROTOCOL

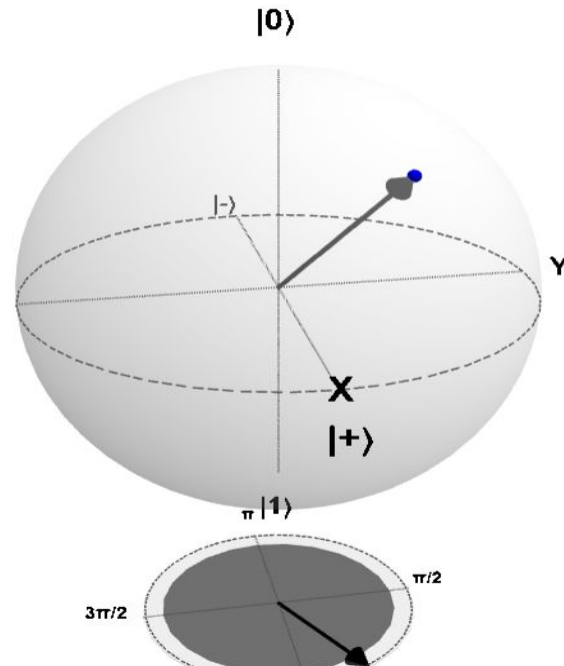


# QUBIT:-

<https://javafxpert.github.io/grok-bloch/>

$$|\psi\rangle = \sqrt{0.85} |0\rangle + (\sqrt{0.15}) e^{i0.15\pi} |1\rangle$$

Prob of  $|0\rangle$



## INTERCEPTION-RESEND ATTACK DETECTION TECHNIQUE:-

The Interception-resend attack detection technique is a method used to detect data breaches in the BB84 key exchange protocol

1. The sender and receiver exchange key bits using the BB84 protocol.
2. A subset of the key bits exchanged between the sender and receiver is randomly selected.
3. The sender and receiver compare the selected key bits to a public reference. The public reference is a set of key bits that are publicly known to both parties and are not part of the key exchange.
4. If the selected key bits match the public reference, it is likely that the key exchange was not intercepted and modified by an eavesdropper. If the key bits do not match the public reference, it is likely that an eavesdropper has intercepted and modified the key.

# IS IT GREENER AND SUSTAINABLE ALTERNATIVE?

It is a mathematical method that is used to ensure the security and integrity of communication channels. While reducing energy consumption and minimizing waste is essential for a sustainable future.

THANK

YOU