

Quantum Key Distribution Simulation with Mininet

Key Components and Flow

1. Network Topology Setup:

- Utilizes Mininet to create a simple network with two hosts (Alice and Bob) connected via a single switch.

2. QKD Simulation:

- Bit and Basis Generation: Alice generates a random sequence of bits and selects random bases (rectilinear + or diagonal ×) for each bit.
- Qubit Encoding: Alice encodes each bit into a qubit based on the selected basis.
- Transmission and Interception:
 - The encoded qubits are transmitted over the network.
 - Eve may intercept and measure the qubits using randomly chosen bases, potentially introducing errors.
- Measurement by Bob: Bob measures the received qubits using his own randomly chosen bases.
- Key Sifting: Alice and Bob publicly compare their bases and retain only the bits where their bases match, forming the shared secret key.

3. Output:

- Displays the keys generated by Alice and Bob.
- Shows the bits intercepted by Eve.
- Provides insights into the presence of eavesdropping based on discrepancies between Alice's and Bob's keys.

4. Mininet CLI:

- After the simulation, the script launches the Mininet Command Line Interface (CLI) for further interaction and inspection of the network.

Step 1: Install Virtualbox and Ubuntu

In the Ubuntu terminal:

```
sudo apt update && sudo apt upgrade -y
```

Enter your password when asked. (pwd- Sarah@2002)

This updates your Linux system.

Step 2: Install Mininet and Git

Run the following in the terminal:

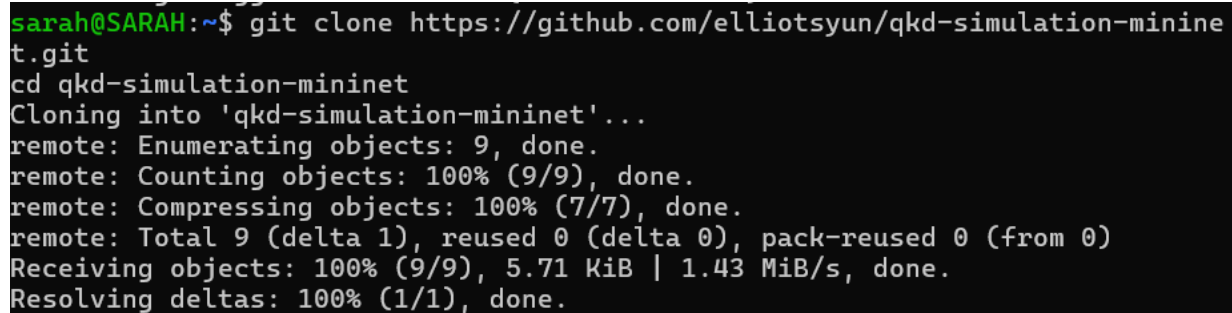
```
sudo apt install mininet git -y
```

Step 3: Install Python and pip

```
sudo apt install python3 python3-pip -y
```

Step 4: Clone the GitHub Repository

```
git clone https://github.com/elliotsyun/qkd-simulation-mininet.git  
cd qkd-simulation-mininet
```



```
sarah@SARAH:~$ git clone https://github.com/elliotsyun/qkd-simulation-minine  
t.git  
Cloning into 'qkd-simulation-mininet'...  
remote: Enumerating objects: 9, done.  
remote: Counting objects: 100% (9/9), done.  
remote: Compressing objects: 100% (7/7), done.  
remote: Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)  
Receiving objects: 100% (9/9), 5.71 KiB | 1.43 MiB/s, done.  
Resolving deltas: 100% (1/1), done.
```

```
ls
```

If you see requirements.txt, install the Python packages: (If not, skip this step)

```
pip3 install -r requirements.txt
```

Step 5: Run the Simulation

```
sudo python3 alice_bob.py
```

This will initialize the network, perform the QKD simulation, and open the Mininet CLI for additional commands or inspection.

```

sarah@SARAH:~/qkd-simulation-mininet$ sudo python3 alice_bob.py
*** Creating network
*** Adding controller
*** Adding hosts:
alice bob
*** Adding switches:
s1
*** Adding links:
(alice, s1) (bob, s1)
*** Configuring hosts
alice bob
*** Starting controller
c0 Cannot find required executable ovs-controller.
Please make sure that it is installed and available in your $PATH:
(/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin)
sarah@SARAH:~/qkd-simulation-mininet$ sudo apt install openvswitch-switch -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openvswitch-switch is already the newest version (3.3.0-1ubuntu3.2).
openvswitch-switch set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
sarah@SARAH:~/qkd-simulation-mininet$

```

ERROR:

This means **Open vSwitch (OVS)**, which Mininet uses to simulate switches, is not installed yet. Then re run the simulation.

```

sudo apt install openvswitch-switch -y
sudo apt install mininet openvswitch-testcontroller -y

```

You should now see the Mininet CLI prompt (usually mininet>). From here, you can test connectivity, run commands, or inspect your simulated hosts:

```

mininet> alice ping bob
mininet> alice ifconfig
mininet> bob python3
mininet> exit()

```

```
sarah@sarah-VirtualBox:~/qkd-simulation-mininet$ sudo python3 alice_bob.py
```

```
*** Creating network
*** Adding controller
*** Adding hosts:
alice bob
*** Adding switches:
s1
*** Adding links:
(alice, s1) (bob, s1)
*** Configuring hosts
alice bob
*** Starting controller
c0
*** Starting 1 switches
s1 ...
Alice Key: [0, 0, 1, 1, 0]
Bob Key: [0, 0, 1, 1, 0]
Eve Bits: [1, 1, 1, 0, 1, 1, 1, 1, 1, 0]
*** Starting CLI:
mininet> alice ping bob
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.87 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.09 ms
a64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.130 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.061 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.071 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.089 ms
alice64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.058 ms
^?^?64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.244 ms
^?^?^?64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.082 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.125 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.077 ms
^C
--- 10.0.0.2 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10236ms
rtt min/avg/max/mdev = 0.058/0.354/1.868/0.557 ms
```

```

mininet> alice ifconfig
alice-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
    inet6 fe80::b4d7:3aff:fef8:d1af prefixlen 64 scopeid 0x20<link>
    ether b6:d7:3a:f8:d1:af txqueuelen 1000 (Ethernet)
    RX packets 44 bytes 4662 (4.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 22 bytes 1888 (1.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mininet> bob python3
Python 3.12.3 (main, Feb 4 2025, 14:48:35) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit
Use exit() or Ctrl-D (i.e. EOF) to exit
>>> exit()
mininet>

```

Alice_bob.py

```

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import Controller, OVSController
from mininet.log import setLogLevel, info
from mininet.link import TCLink
from mininet.cli import CLI

import random

# BB84 Quantum Key Distribution Functions
RECTILINEAR = '+'
DIAGONAL = 'x'

def generate_bits_and_bases(length):
    bits = [random.randint(0, 1) for _ in range(length)]
    bases = [random.choice([RECTILINEAR, DIAGONAL]) for _ in range(length)]

```

```

return bits, bases

def encode_bits(bits, bases):
    return [(bit, base) for bit, base in zip(bits, bases)]

def intercept_and_measure(encoded_bits):
    guessed_bases = [random.choice([RECTILINEAR, DIAGONAL]) for _ in
encoded_bits]
    measured_bits = []
    for (bit, base), guessed_base in zip(encoded_bits, guessed_bases):
        if base == guessed_base:
            measured_bits.append(bit)
        else:
            measured_bits.append(random.randint(0, 1))
    return measured_bits, guessed_bases

def measure_bits(encoded_bits, bob_bases):
    bob_measured_bits = []
    for (bit, base), bob_base in zip(encoded_bits, bob_bases):
        if base == bob_base:
            bob_measured_bits.append(bit)
        else:
            bob_measured_bits.append(None)
    return bob_measured_bits

def qkd_simulation(length=10):
    alice_bits, alice_bases = generate_bits_and_bases(length)
    encoded_bits = encode_bits(alice_bits, alice_bases)
    eve_bits, eve_bases = intercept_and_measure(encoded_bits)
    bob_bases = [random.choice([RECTILINEAR, DIAGONAL]) for _ in range(length)]
    bob_measured_bits = measure_bits(encoded_bits, bob_bases)
    sifting_indices = [i for i in range(length) if alice_bases[i] == bob_bases[i]]
    alice_key = [alice_bits[i] for i in sifting_indices]
    bob_key = [bob_measured_bits[i] for i in sifting_indices if bob_measured_bits[i] is not
None]
    return alice_key, bob_key, eve_bits

class MyTopo(Topo):
    def build(self):
        alice = self.addHost('alice')
        bob = self.addHost('bob')
        switch = self.addSwitch('s1')
        self.addLink(alice, switch)

```

```
self.addLink(bob, switch)
```

```
def run():
```

```
    topo = MyTopo()
```

```
    net = Mininet(topo=topo, controller=OVSController, link=TCLink)
```

```
    net.start()
```

```
    alice, bob = net.get('alice', 'bob')
```

```
    alice_key, bob_key, eve_bits = qkd_simulation()
```

```
    print("Alice Key: ", alice_key)
```

```
    print("Bob Key: ", bob_key)
```

```
    print("Eve Bits: ", eve_bits)
```

```
    CLI(net)
```

```
    net.stop()
```

```
if __name__ == '__main__':
```

```
    setLogLevel('info')
```

```
    run()
```