Önálló Laboratórium beszámoló

Radnai Barna

Absztrakt

Az általam megvalósított projekt egy Python program, aminek a segítségével két, a föld tetszőleges pontján levő, földi állomás közötti titkos kommunikációt modellezek. A titkosítást kvantumkommunikációra alkalmas műholdak segítségével valósítom meg, a BB84 protokollt használva. A modell figyelembe veszi kommunikációs csatorna zavaró tényezőit, a kommunikációhoz szűkséges időt, azt, valamint hogy a tényezőkkel együtt létrehozható-e a kulcscsere a földi egység és a műhold között. Emellett támadókat is modellez a műhold-földi állomás linken, valamint a földi állomások közötti linken, ezeken pedig ismert támadási modelleket képes végrehajtani.

A modellezett folyamatról pár mondat

Állandó vita szokott lenni szakértői körökben, hogy a kommunikáció középpontjában álló egység (jelen esetben a műhold) úgynevezett *Trusted Node,* vagy *Untrusted Node* típusú legyen-e. Az első eset azt jelenti, hogy az egység tartalmaz olyan információkat, amikhez ha egy rosszindulatú támadó (innentől kezdve erre a szereplőre *Eve*-ként hivatkozok) hozzáfér, azok segítségével visszafejthető lesz a kommunikáció egy része, vagy teljes egésze. A második pedig azt, hogy az egység nem tartalmaz olyan információkat, amik segítségével a titkosítás feltörhető lenne.

Már ez a leírás sejteti, hogy ha pusztán az elméleti oldalát nézzük a dolgoknak, akkor a *Trusted Node* az evidens választás. Azonban praktikus okokból sokszor mégis az *Untrusted Node* típusú megoldást választják.

Ebben a dolgozatban én is egy *Untrusted Node* típusú modellt dolgoztam ki, ennek okai a következők:

- A nyalábszélesedés jelensége downlink csatornán nagyságrendekkel kevésbé érvényesül, mint uplink csatornán, ebből kifolyólag a hatékony kommunikáció érdekében érdemesebb a műholdon generálni a qubiteket.
- Bár az összefonódott qubit-pár egyik fele a műholdon van, annak “megfejtéséhez” fizikailag hozzá kell férni, vagyis ahhoz, hogy valaki közvetlenül lehallgassa a kommunikációt, fizikailag csatlakoznia kellene a műholdhoz. A világ és a tudomány jelenlegi állása szerint azonban erre kevés esély van.

A program működésének leírása

A program egy műhold mozgását modellezi. Az útvonal 4 szakaszra osztható

1. A műhold kezdőpontja és az első földi állomás által látott szakasz kezdőpontja közötti szakasz
2. Az első földi állomás által látott szakasz
3. Az első földi állomás végpontja és a második szakasz kezdőpontja közötti szakasz
4. A második földi állomás által látott szakasz

Az első szakasznál a műhold nem végez semmilyen kommunikációt, csak szimplán végighalad rajta.

A második szakasznál a műhold és a földi bázis között BB84 szerinti kulcscsere történik.

A harmadik szakasz az elsőhöz hasonlóan csak egy kommunikációmentes út.

A negyedik szakasznál a műhold és a második földi bázis között szintén végrehajtódik a BB84 protokoll, majd ennek sikeressége után a frissen létrejött kulcs segítségével a műhold elküldi ennek az állomásnak az első állomás kulcsát. Ezután a két földi állomás már rendelkezik annyi információval, hogy biztonságosan létre tudnak hozni egy közös, tetszőleges szimmetrikus kulcsot (Jelen esetben ez a kulcs egy AES-256 kulcs).

A program első lépésben kiszámolja a pálya, valamint az egyes szakaszainak hosszát. Ezekhez a következő adatokra van szűksége:

- Az állomások, valamint a műhold kezdőpontjának szélességi és hosszúsági koordinátáira
- A földi állomások zenit szögeire
- A műhold keringési pályájának magasságára

Ezeket az adatokat felhasználva először egy függvény segítségével kiszámoltam a műhold kezdőpontja és az első állomás, valamint a két állomás közöti távolságot, majd egy másik függvény segítségével azt, hogy milyen hosszúak az állomások által látott zónák, valamint ezeknek kezdő- és végpontjait.

A függvények:

```python
def CalculateDistance(lat1, lat2, lon1, lon2, satelliteheight): #két földi állomás közötti távolság kiszámítása a műhold magasságát figyelembe véve
    lat1=math.radians(lat1)
    lat2=math.radians(lat2)
    lon1=math.radians(lon1)
    lon2=math.radians(lon2)
    dlon=lon1-lon2
    dlat=lat1-lat2
    a=math.sin(dlat/2)**2+math.cos(lat1)*math.cos(lat2)*math.sin(dlon/2)**2
    c=2*math.asin(math.sqrt(a))
    r=6371+satelliteheight
    return (c*r)

    #source: https://www.geeksforgeeks.org/program-distance-two-points-earth/

def GetVisibleZone(g1, distance_from_startpoint, height_of_satellite): #visszaadja a földi állomás látható zónájának kezdő- és végpontját
    alpha=(360-g1.zenit)/2
    alpha=math.radians(alpha)
    a=6371+height_of_satellite
    f=6371
    b=math.sin(alpha)
    c=b*float(f)/float(a)
    beta=math.asin(c)

    beta=beta*180/3.141
    gamma=360-2*(alpha*180/3.141+beta)
    l=(gamma/float(360))*2*a*3.141
    points=[]
    points.append(distance_from_startpoint-(l/2))
    points.append(distance_from_startpoint+(l/2))
    return points
```

Ha ez megvan, 5 ponttal és a teljes út hosszával rendelkezünk. Ezek alapján létre lehet hozni a fentebb említett négy szakaszt.

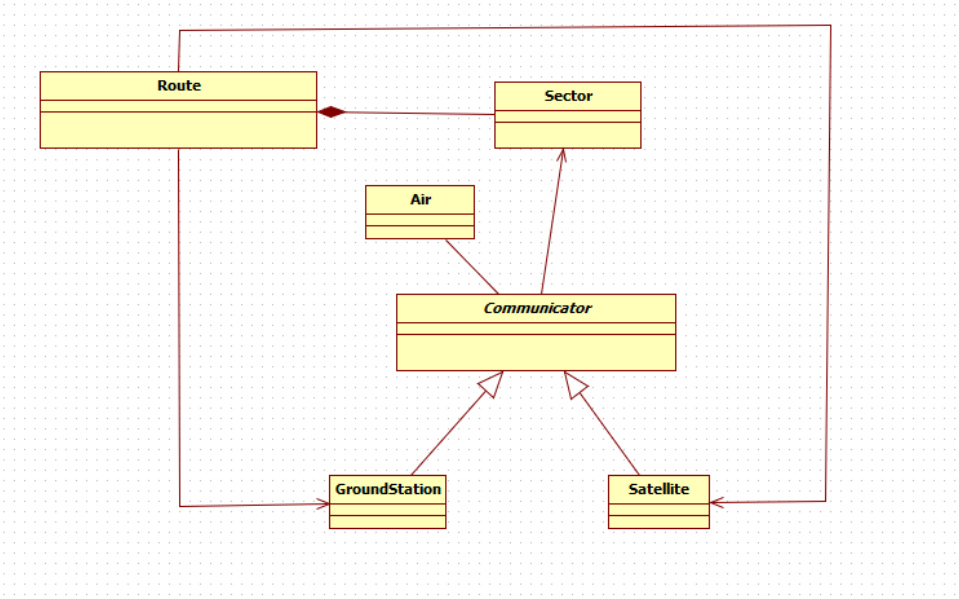
Ezek után el lehet kezdeni a szimulációt.

A szimuláció lényegi elemei a kommunikáló felek. Ezek a műhold és a földi bázis, de ezek tovább bővíthetőek, hiszen közös ősosztályból vannak származtatva.

A kommunikáló feleknek két interfészük van: egy rádiós és egy optikai. Az optikai interfészen végzik a kvantumbittekkel kapcsolatos műveleteket, az optikai csatornán pedig az egyéb dolgokat egyeztetik. A műhold képes kvantumbitek generálára, valamint azok küldésére, a földi bázis pedig a vételükre.

A program osztályainak leírása

Osztálydiagram:



Route:

Ez az osztály kezeli midnen mást. Elsőnek kiszámolja és részekre osztja az útvonalat, majd ezután végrehajtja a szimulációt

Függvényei

- Calculate()

  Kiszámolja az útvonalat, létrehozza a szektorokat

- Simulate()

  Ez a függvény hajtja végre a szimulációt. Futása során meghívja az adott szereplők függvényeit kronológiai sorrendben.

- BB84()

  Ez a függvény a szereplők azon függvényeit hívja meg, amik a BB84 protokollhoz kapcsolódnak.

- Reconcilation()

  Ennek a függvénynek a szerepéről egy későbbi fejezetben részletesebben írok majd.

Sector:

A fentebb említett szakaszokat reprezentáló osztály. Tartalmazza a hosszukat, a megtételükhöz szükséges időt, valamint a kezdő- és végpontjaikat.

Communicator:

Absztrakt osztály, a műhold és a földi bázis ősosztálya, ezeknek a közös tulajdonságait és függvényeit tartalmazza.

Függvényei

- SendRadioBroadcast()

  A rádiós interfészen való üzenetküldést modellező függvény. Paraméterként egy listát vár el, amelynek első eleme az üzenet típusa, többi eleme pedig az üzenet tartalma.

- RecieveRadioBroadcast()

  A rádiós interfészen való üzenetvevést modellező függvény.

GroundStation:

A földi állomást modellező osztály. Rendelkezik egy látószöggel, valamint az egyik tulajdonsága az, hogy a kommunikációban az első, vagy a második földi bázis-e. Ettől függően egy kicsit másképp viselkedik

Függvényei

- Makekey()

  A végleges kulcs létrehotásáért felelős függvény. Ha a második földi bázis, akkor annyiban viselkedik másképp, hogy a műholdtól megkapja az első állomás kulcsát XORolva a saját kulcsával, majd ezt az üzenetet XORolja a saját kulcsával. Ha az első, akkor pedig a BB84 során létrejött kulcs első 256 bitjét veszi.

- RecieveQubit()

  A qubit elkapását végző függvény. Ha nem érkezik meg a vár qubit, akkor ezt jelzi a műholdnak, ha megérkezik, akkor megméri.

- MeasureQubit()

  A konstruktorban kapott qubitet megméri a konstruktorban kapott bázissal, majd az így kapott bitet és a használt bázist eltárolja.

- SendMeasureBases()

  A BB84 egyik lépését végzi el, a rádiós csatornán elküldi a mérés során használt bázisokat

- CheckParity()

  A helyesen mért bitek egy részhalmazának paritását ellenőrzi. Ha egyesből van több nullát, ha nullásból van több, egyet ad vissza. Ennek a függvénynek

a reconcilation során lesz jelentőssége, amit később fogok részletesebben kifejteni.

- Reconcilation()

  Később fejtem ki ennek a lényegét

- PrintInfo()

  Információk kiiratása

Satellite:

A műholdat modellező osztály. Rendelkezik egy tengerszint feletti magassággal és egy sebességgel.

Függvényei

- CheckParity()

  Összeveti a földi bázis által mért paritást a kulcs ugyanazon részhalmazával, amit ő is megmért. Erről bővebben később.

- CompareLists()

  Két lista tartalmát veti össze, a megegyező részeinek indexeit visszaadja egy listában. A BB84 protokoll egyik lépésében van szerepe.

- SendMatchingBases()

Rádiós csatornán elküldi azoknak a bázisoknak az indexeit, amik megegyeztek. A BB84 protokoll egyik lépésében van szerepe.

- GenerateQubit()

  Egy qubit generálásaért felelős osztály. Visszatérési értékként egy Photon típusú objektumot ad.

- SendQubit()

  A qubitküldést menedzselő osztály. Képes kezelni azt, ha egy qubit elveszik az éterben a kommunikáció közben. Ennek heurisztikájáról később bővebben is beszélek.

- SendAmountOfQubit()

  A konstruktorban megadott számú qubitot üldi ki az optikai interfészen. Visszatérési értékként azt a mennyiséget adja, ahány qubit elveszett az éterben

- SendXORedKey():

A rádiós csatornán kiküldi az első és a második földi bázis kulcsának az XOR-olt értékét.

Air:

A műhold és a földi bázis közötti közeget modellező osztály. Ezen keresztül haladnak mind az optikai, mind a rádiós csatornán küldött üzenetek. Tulajdonságai közé tartozik a transzmittancia, ami a qubit éterben való elveszésének a valószínűsége, tartozik hozzá egy műhold és egy földi állomás, opcionálisan tartalmazhat egy támadót, valamint tartalmazza, hogy a csatornában épp milyen évszak van és milyen időjárás.

Függvényei

- SetTransmittancy()

  Beállítja a transzmittancia értékét. Ez számos tényezőtől függ, többek között az évszaktól, az időjárási viszonyoktól, a műhold magasságától, a lézernyaláb szélesedésétől, stb. Az érték pontos kiszámításához *Mihály András atmosModell.py* nevezetű programját használtam fel, valamint G*alambos Máté, Műholdas kommunikáció modellezése és vizsgálata kvantuminformatikai alkalmazásokhoz* című TDK dolgozatát használtam az elméleti háttér megértéséhez. Innen is köszönet nekik!

- SendQubit()

  A qubit légkörben való haladását modellező függvény. A transzmittancia értékétől függően vagy elveszetnek nyilvánít egy qubitet, vagy továbbítja azt a földi állomásnak

- BroadcastSatelliteToGround()

  A műholdtól érkezett, rádiós üzenetet továbbítja a földi állomásnak

- BroadcastGroudToSatellite()

A földi állomástól érkezett, rádiós üzenetet továbbítja a műholdnak

- IsSecond()

Visszaadja, hogy a hozzá tartozó földi állomás a kommunikációban első, vagy második számú-e.

Photon:

A polarizált fotont modellező osztály. Működése körvonalakban hasonlít egy igazi, a kvantumkommunikációban használt, polarizált fotonhoz: ha olyan bázisban mérik meg, amiben polarizálva lett, akkor visszaadja tényleges értékét, azonban ha más bázisban mérik, akkor véletlenszerűen ad 0, vagy 1 értéket.

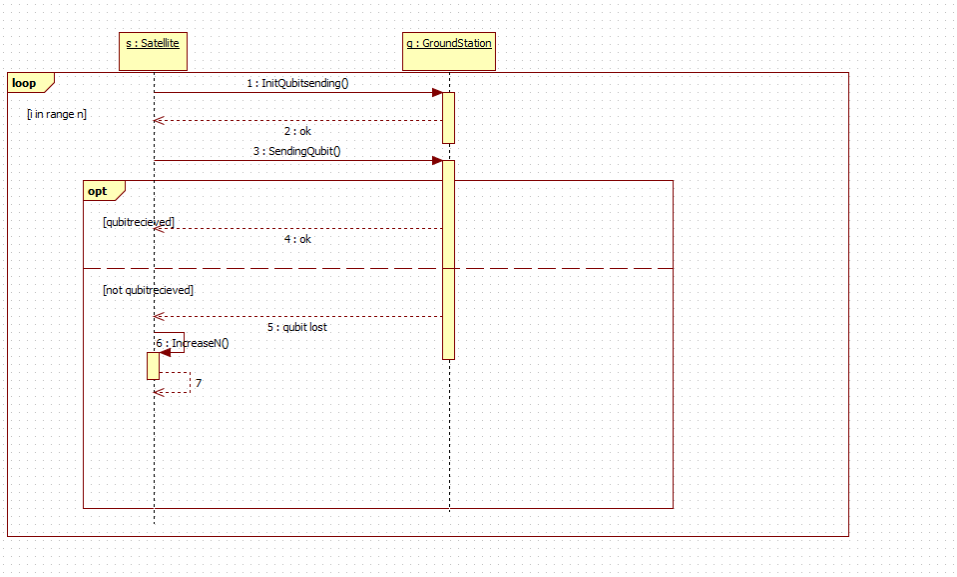
Függvényei

- Measure()

Ha a konstruktorban kapott bázisban lett kódolva, akkor visszatéríti az értékét, ha nem, akkor egy véletlenszerű bitet ad vissza.

Transzmittanciakezelés

A légkört modellező osztály úgy van megírva, hogy a transzmittancia értékétől függően elveszhet a rajta átmenő foton. Ennek kezelésére a következő heurisztikát találtam ki és alkalmaztam:



Röviden a lényeg: A műhold minden egyes qubit küldése előtt küld egy inicializáló üzenetet, ha a földi bázis ezt elfogadja, akkor küldi a qubitot is. Az inicializáló üzenetben a csatorna minőségét figyelembe véve egyeztetnek egy időkorlátot, amin túl a földi állomás visszaszól, ha nem érkezett qubit, majd a földi bázis eggyel megnöveli a még küldendő qubitek számát. Ha pedig megérkezett, akkor ezt jelzi a műholdnak.

Ebben a megoldásban komoly tényező lehet az is, hogy mi történik akkor, ha Eve próbál meg ilyen rádiós üzenetet küldeni. Erre a megoldás az autentikáció. A hétköznapokban használt autentikációs megoldások, algoritmusok nagy többségben tartalmaznak nyilvános kulcsú elemeket is, ami számunkra nem feltétlenül ideális, hiszen a kvantumkommunikáció egyik fő előnye, hogy ezeket az infrastruktúrákat elkerüli. Éppen ezért érdemes lenne egy hosszú távú, biztonságosan cserélhető kulcsú, szimmetrikus kulcsú autentikációs megoldást alkalmazni. Ennek a feltárására ebben a dolgozatban nem kerítettem sort, azonban egy TDK dolgozat, vagy egy szakdolgozat keretében ebbe az irányba is kitekintést lehet venni.

Az elméleti BB84 protokoll tesztelése, hibái, megoldásai

A BB84 egy ideális, kvantumos mérő- és adóeszköz nélküli Eve-vel a következő lépésekkel is biztonságosan működhet:

1. A műhold és a földi bázis megegyezik két, egymással ortogonális polarizációs bázisban (ez a megegyezés akár a legyártásukkor is megtörténhet) .
2. A műhold generál egy random értékű fotont, majd ezt a két, előre egyeztetett bázis egyikével polarizálja és elküldi a földi állomásnak. Ezt megismétli n-szer.
3. A földi bázis minden érkező fotont eltárol, majd ezeket egy-egy, véletlenszerűen választott bázissal megméri.
4. A földi állomás a rádiós csatornán elküldi a műholdnak az általa használt bázisokat a műholdnak
5. A műhold összeveti ezeket az általa használt bázisokkal, eldobja azokat a biteket, amelyeket különböző bázisokban mértek és visszaküldi azokat, amik megegyeztek
6. A földi állomás ez alapján szintén eldobja a rossz bázisban mért biteket. Ezek után létrejött a közös kulcs

A nagy számok törvénye alapján körülbelül a használt bázisok fele rossz volt, így a az úgynevezet Qber ráta nagyjából 0.5 körül lesz.

A műhold ezután a második állomással is végrehajtja a protokollt, majd közli vele az első állomás által küldött kulcsot is. Ez után a két földi állomás között is létrejött a titkos kulcs:

```
New York
végleges kulcs:  [1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1,
0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0,
0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1]
------------------------------
Budapest
végleges kulcs:  [1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1,
0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0,
0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1]
```

A következő szimulációban beállítom, hogy a csatornában legyen jelen Eve.

```
groundstation_1=comm.GroundStation(lat=lat1, lon=lon1, zenit=45, IsSecond=0)
groundstation_2=comm.GroundStation(lat=lat2, lon=lon2, zenit=45, IsSecond=1)
satellite=comm.Satellite(lat=0.0, lon=0.1, height=500, speed=8)
groundstation_1.AddName("New York")
groundstation_2.AddName("Budapest")
satellite.AddName("Satellite-1")
air_1=comm.Air(name="New York Air", IsEve=1, season="summer", weather="hazy")
air_2=comm.Air(name="Budapest Air", IsEve=1, season="summer", weather="hazy")
```

Ekkor a következőt tapasztalom:

```
New York
végleges kulcs:  [1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1,
1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1,
0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1]
------------------------------
Budapest
végleges kulcs:  [1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1,
1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1]
```

Az látszik, hogy a két bitsorozat nem egyezik. Ennek az oka a következő: Eve jelen van támadóként és minden egyes elküldött qubitet elkap, majd megmér és a mért értéket egy

véletlenszerű bázisban kódolja tovább. Bár a műhold és a földi állomás tökéletesen leegyeztették egymás között a használt bázisokat, a az értékük mégis 50%-ban el fog térni, hiszen Eve belekontárkodott a kommunikációba és mivel a nagy számok törvénye itt is érvényes, ezért a bitek fele nem lesz érvényes.

Ennek a jelenségnek az orvoslására találták ki a reconcilation nevű lépést. Ez tulajdonképpen annyit tesz, hogy minél kevesebb információ közzétételével, minél biztosabban meg tudják azt állapítani, ha Eve jelen volt a csatornában.

Erre a legpraktikusabb megoldás az, ha a létrejött kulcs egy bizonyos alterének a paritását megvizsgálják és összeegyeztetik. A megoldásomban nem jártam utána ennek a lépésnek a legoptimálisabb alkalmazásának, így teszt demonstráció jelleggel kiválasztok 3 darab, 10 elemű részhalmazt és azokat egyeztetem össze. Nagy valószínűséggel már ebből is kiderül, ha Eve jelen volt.

```
kulcs mérete:  336
összesen elküldött qubitek száma:  700
Qber:  0.48
Eve valószínűleg nem volt jelen
Eve valószínűleg jelen volt
Eve valószínűleg nem volt jelen
```

A műhold Reconcilation() nevű függvénye írja ki a következőket. A fenti példa jól mutatja, hogy miért érdemesebb több alteret is megvizsgálni: Látszik, hogy csupán egy alkalommal jelezte azt, hogy valószínűleg Eve jelen volt.

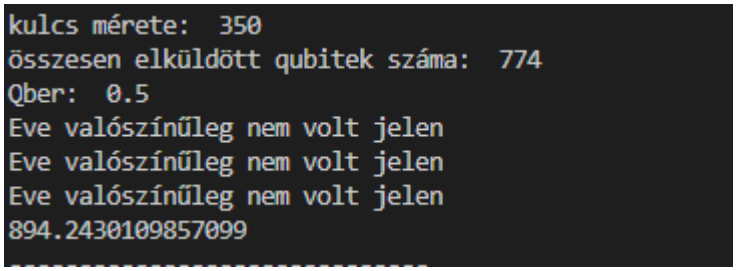
A hátulütője annak, ha sok altér paritását egyeztetjük össze az az, hogy Eve ezzel értékes információhoz juthat a létrejött kulcsokról, ha esetleg csak a nyilvános csatornát figyeli, akkor is.

Ha teszteljük Eve nélküli csatornán is a reconcilationt, akkor a következő eredményt kapjuk:

```
kulcs mérete:  355
összesen elküldött qubitek száma:  700
Qber:  0.5071428571428571
Eve valószínűleg nem volt jelen
Eve valószínűleg nem volt jelen
Eve valószínűleg nem volt jelen
```

A protokoll működése nagy transzmittancia esetén

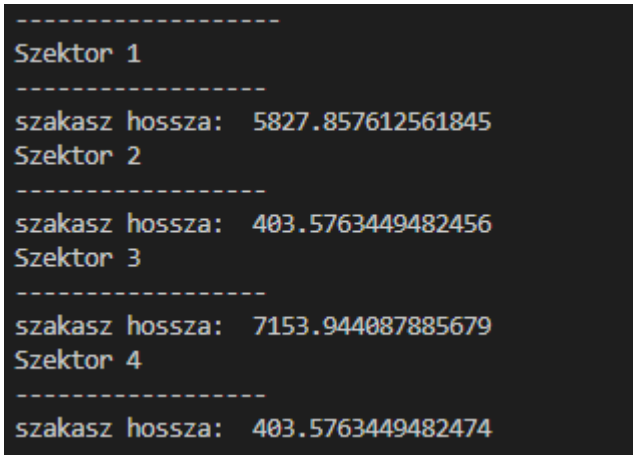
Egy előző fejezet során már kifejtettem, hogy ha a qubit elveszik a térben, arra hogy reagál a rendszer. Nézzük most meg gyakorlatban:



A transzmittanciát manuálisan 0.7-re állítottam és a cél az volt, hogy 700 qubit sikeresen átmenjen a csatornán. A képernyőképen látszik, hogy ehhez 774 qubit elküldésére volt szűkség.

Szektorok

Egy korábbi fejezetben már írtam a szektorok szerepéről, valamint létrehozásáról. A következő képen látszik, hogy eredményesen kiszámolja az egyes szakaszok hosszát a program:



Továbblépési lehetőségek

Mint azt korábban említettem, érdemes lenne kidolgozni egy szimmetrikus kulcsú autentikációs protokollt a rádiós, nyilvános üzenetek autentikálására, esetleg titkosítására

Emellett a rendszer jelenleg csak egy műholdat és két földi állomást tud modellezni és csak downlink csatornán. Érdemes lehet megvizsgálni egy esetleges uplink csatornát, valamint azt is megvizsgálni, hogy több műholddal hogyan lehetne egy hasonlóan működő rendszert létrehozni.

Források

Galambos Máté: Műholdas kommunikáció modellezése és vizsgálata kvantuminformatikai alkalmazásokhoz, 2009

Mihály András: atmosModell.py