BB84: Write a hidden secret in the quantum world

Ohad Forman[1,2] , Noam Paryanti[1,3] and Georgi Gary Rozenman[1,2]
[1] *Raymond and Beverly Sackler School of Physics and Astronomy, Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel*
[2] *School of Electrical Engineering, Iby and Aladar Fleischman Faculty of Engineering, Tel Aviv University, Tel Aviv 69978, Israel*
[3] *School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel*
(Dated: June 4, 2022)

We observe the results of the BB84 protocol Analog using a pulsed laser signals as the hardware systems that required for QKD. A simulation results of the chain of communication, the probabilities of Errors from Eavesdropper and the probabilities to detect Eavesdropper with respect to the Test string size is shown.We show how A GUI based on the simulation can deploy and help a "End user" understand and use the protocol for encrypted communication without deep knowledge in Quantum physics and mathematics and can use it when the technology will be available for the "end user"

I. INTRODUCTION

Cryptography, from Ancient Greek can translate to "To write a hidden secret", since ancient times the need to secure information and choose the authorized sides to the information was essential in a way that a unauthorized side can't accesses the hidden information. The definition of Cryptography is the constructing and analyzing of protocols that allow to two or more authorized sides to exchange information without the ability of unauthorized side to get access to the information. In this era the transformation of secret data is essential to mankind. health, financial and state security data is only a part of the hidden information that need to be encrypted for a safe communication, in this era most of the communication is online and also stored in a cloud based technology. The Quantum World meets the Cryptography World in two main roots. One root is the Evesdropper root, where in the close future a Evesdropper will be able to evedrop a encrypted information with the help of Quantum computer[1]. In the side of the authorized sides of communication, The BB84 protocol introduced by Bennett and Brassard in 1983 [2] make a description of how to use rules of the Quantum mechanics to the purpose of generate a Quantum Key distribution protocol.

At 1900 quantum hypothesis made by Max Planck[3]. that any energy-radiating atomic system can theoretically be divided into a number of discrete "energy elements" such that each of these energy elements is proportional to the frequency with which each of them individually radiate energy.
5 Years later at 1905, with the motivation to explain the photoelectric affect, A.Einstein made the hypothesis that the light itself is made of individual quantum particles[4], which in 1926 came to be called photons by Gilbert N. Lewis[5]. The state in quantum physics is the mathematical description of the quantum physical properties of a quantum in the shape of the probabilities to measure the outcomes of each possible measurement on a system.
In 1939 P. A. M. Dirac introduce a new notation to quantum mechanics[6], Based on the Bra-Ket. One can say that a notation is only a notation, but Dirac notation made the quantum state concept wider and dipper in the understanding of the quantum state nature.
Jumping to 1970, J.L.Park build the foundations to the "no cloning theorem"[7].the no-cloning theorem states that it is impossible to create an independent and identical copy of an arbitrary unknown quantum state.

With this principals from the quantum mechanics world the BB84 protocol is able to solve a security problems of the classical encryption algorithms. The BB84 algorithm using the "no cloning theorem" in the shape of testing for the existing of a evesdropper, if a evesdropper is able to copy the encrypted information without changing it he cannot be discover, this problem has no solution in the classic world, this is the main problem that being solved with the BB84 protocol. another problem in the classical world is the difficulty in making a random Bit, that is essential to encryption, the states in quantum mechanics can be a superposition of 2 states with 0.5 probability to measure each one, by that this is a true random Bit generator if we call one of the states '0' and the second '1'.
The protocol will hold secure in the era of Quantum computers, since it is not depend on the computing power of the evesdropper and depend only on the principles of the quantum mechanics world.

A. Random in technology

random result of a process is a result sample that distribute equally in the result field, assume that the results field is [0,1], as a binary bit. a random result is a function that return 0 or 1 with equal probability's 0.5. achieving such a function is not trivial, a computer software depends on a defined set of actions to deliver a result, an algorithm. A can be only "pseudo random" and breathable because of that, real randomness is non invariable function, and a "pseudo random" is reversible, one can try and succeed to understand the way the algorithm operate.
In the nature we witness a random phenomenons, in the Quantum world, particle decay process's and more, we

can use this phenomenons to achieve a ideally random generator.

B. One Time Pad

The One Time Pad is an Classical encryption method. the method is to generate a security key with logical binary bits, and share the key secretly between authorized sides of the communication, the key binary string must be random sequence, now a message with the same length is coded in binary representation. A Binary operator of 2 bits applies to a key bit and a massage bit result a random result also that the sender can sent safely, the receiver can apply the same logical operator with the result and the key to get the un-encrypted message XOR operator meet the condition for that operator with the Truth table:

BIT A	BIT B	XOR(A,B)
0	0	0
0	1	1
1	0	1
1	1	0

Table I. XOR truth table

fully secure in principle precede can achieve under 3 condition:

- The key is fully random
- The key can be used for one time only
- The key known only to authorized sides of communication

C. Photon Polarization

The mane protocol in quantum cryptography - BB84 requires from the information bits to have some quantum properties. Since polarized Photons are stable, a practical way of transmission and they have the right quantum properties to execute BB84, they are the mane implementation of the protocol.
When classical polarized light is polarized along the X axis and then passes through a half-wave plate where the angle between the polarization of the electromagnetic wave and the mane axis of the half-wave plate is θ the polarization is rotated by 2θ, as described in figure 1.

In photons there is a similar phenomenon, when a photon passes through a half-wave plate were the angle between the polarization of the photon and the mane axis of the half-wave plat is θ the polarization is rotated by 2θ. So one can use a half wave plate to rotate the polarization of a linearly polarized photon. Beam splitter is a device that lets light to pass through it or to be reflected based on the polarization of the light. For example a beam splitter can ignore linearly polarized light in some angle we can define as 0, and completely reflect linearly polarized light in angles 90,-90. In this case a photon with some polarization angle θ will pass with a probability of $cos(\theta)^2$ or be reflected otherwise, this is similar to Malus' low for polarizers.

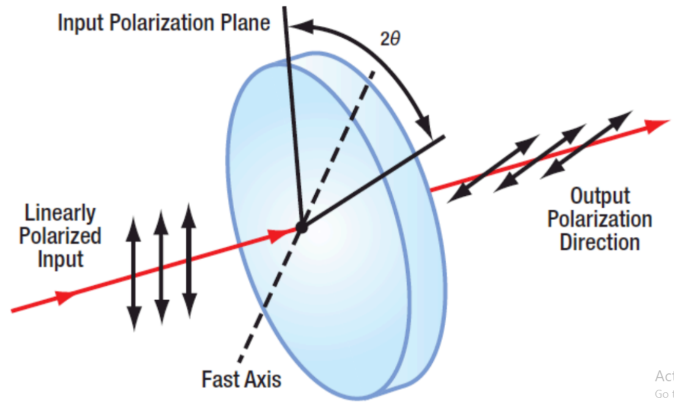


Figure 1. visual description of half-wave plate

D. Measurement Operators - Dirac notation

Armed with a basic understanding of photon polarization we can write a mathematical description of the phenomenon.
denote the state $|\theta>$ as the state that represents the polarization in the direction of θ. Generally a state $|\theta>$ can be described as a super position of a base with two orthogonal angels, for example -45°,45° and 0°,90°. We know that they are in fact orthogonal because the polarization vector of a photon polarized in the direction of one angle does not have a component in the direction of the second one:

$$< -45|45 >=< 0|90 >= 0 \tag{1}$$

Because the probability of a polerized photon to pass though a beam splitter (or measuring it in one specific base) is controled only by the angle of the polarization. The transition between the bases is given by:

$$|0 >= \frac{1}{\sqrt{(2)}}|45 > + \frac{1}{\sqrt{(2)}}| - 45 > \tag{2}$$

$$|90 >= \frac{1}{\sqrt{(2)}}|45 > - \frac{1}{\sqrt{(2)}}| - 45 > \tag{3}$$

the 0,90 base is called "+" and the -45,45 base is called "X".
To measure the the different states we can define the follwing operators:

$$|M_x >= |45 >< 45| - | - 45 >< -45| \tag{4}$$

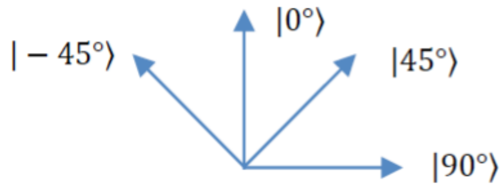


Figure 2. visual description of the 0-90 plane

$$|M_+ >= |0 >< 0| - |90 >< 90| \tag{5}$$

So measuring a state in the same base as the operator has the same result every measurement, but measuring in a different base results a random result every time. This is the most important quantum property in BB84 because all of bob's (and eve's) measurements are done by choosing between M_x and M_+ and 0 is represented by 90 or 45 and 1 is represented by 0 or -45 depending on the base that was chosen.
Bob will use a system of a beam splitter and a half-wave plate to implement M_x and M_+ he would let the photon pass the half-wave plate to measure in the 0,90 base and rotate the polarization to measure -45,45.

E. BB84 Protocol

BB84 Protocol[2] is the first public distributing key protocol that use the Quantum mechanics principals to made a safe exchange of a secret key that can be used with the One Time Pad method. If the protocol execute in order and with the appropriate technology(Appendix: technologies) is fully secure and can detect a Evesdropper, what can't be done in classic protocols.

1. BB84 Protocol steps

Eve steps, as a representation of a Evesdropper activity

1. Alice and Bob generate a random bits array in the size of the binary representation of the key+test+safe margin wanted for the bases. Alice generate a array in the same size for the bits
   Eve generate the same as Alice
2. Alice code the bits in the different bases (+ or x) as every photon represents one bit.
   Eve measure the state sent by Alice, with the bases generated.Eve code the bits in the bases and send this state with a single photon.
   Bob measure the state sent by Eve
3. Alice and Bob publicly compare the bases, discard bits with different bases and save bits with the same bases.
   Eve also intercept this public information
4. Alice and Bob chose a size of test string and the location in the array of binary data to compare. if the comparison test result in different bit the protocol is stooped and Eve is discovered.
   If the there is no mismatches Alice proceed to the next step.
5. Alice send the message with One Time Pad method
   Eve intercept the encrypted message, now she have a information about the bases chosen that was matched between Alice and Bob and can use algorithms to brake the encryption and not discover by Alice and Bob.
   Bob received a message that can be gibberish or Alice message, but in the two cases not encrypted (after decryption using One Time Pad method)

If Eve is not present the test results show that. Alice and Bob use the key generated in One Time Pad method safely.

2. BB84 Protocol Probabilities

We define events of the protocol as follow:

ESB := Eve chose a base equal to Alice
BSB := Bob chose a base equal to Alice

If Eve chose the same base as Alice there will be no change in the state and Eve is not detectable

ESB as a probability of 0.5
BSB as a probability of 0.5
and independent

The Event that correspond to mismatch between the bits of Alice and Bob when chose the same base is the important event, in this event Eve can be discovered.

$$OBE := P\big(\overline{ESB}, BSB\big) = \frac{1}{2} \cdot \frac{1}{2} = 0.25 \tag{6}$$

From that the probability of mismatch in the bits where the bases are equal is 0.25
Now we define the event successful discovery of Eve in the test step:

ST := test is success in finding Eve when present

The probability to NOT find Eve in the test step is the independent events where of every test bit is the event $\overline{OBE}$. the probability to NOT detect Eve is the Event where every test bit is match between Alice and Bob

$$\overline{ST} := P(\overline{OBE})^N$$

where N = Test string size
The complimentary event is the event of Eve being detect, we defined this probability as a confidence level in the test step:

$$Confidence\ level := ST = 1 - P(\overline{OBE})^N = 1 - 0.75^N \tag{7}$$

let CF(N) be the confidence function as function of N

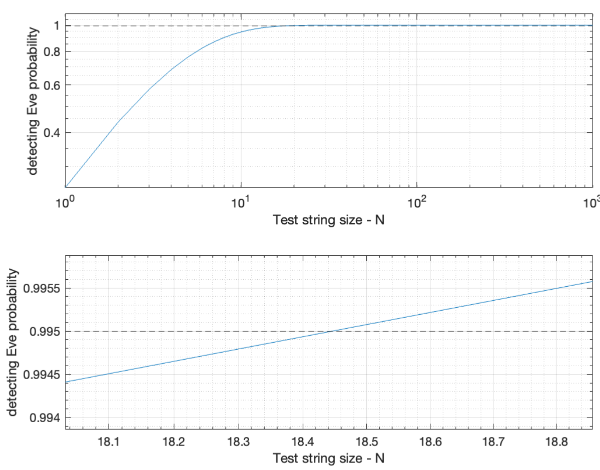


Figure 3. CF(N) - a Y=0.995 line (- -) represent a confidence level of 0.995, CF(N) squared up meet this level at N=19. CF converge relative "fast" to 1, where relative "fast" is considered with test string that make the computing "expensive" in time and resources

II. EXPERIMENT CONFIGURATION

The hardware of the experiment is A pulsed laser transmitter, photons detectors,beamsplitter, Half wave plate. We introduce their symbols (Table II). and represent the experiment configuration in a block diagrams.

- Pulsed laser - produces a linearly polarized photons pulses - The initial quantum state.
- Half wave plate - rotate the polarization the rotation of the plate is half of the polarization rotation (Appendix-Half wave plate)
- Beam splitter - detect and distinguish a super position of the state, if the pulsed laser passes the half wave plate with a wrong bases the beamsplitter will transmit 0.5 (ideally) of the photons to every detector. with the same bases all the photons will transmit to the right detector.
- Laser pulses detectors - a system that detect photons, include 2 detectors, if one is getting photons a led will indicate that detection, if both detect light in approximately same intensity only 1 will indicate by the led.

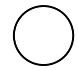
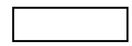
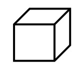
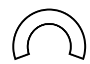
Device	*Symbol*
Half Wave plate	
Pulsed laser	
Beam spliter	
Photons detecter	

Table II. Symbols of the Hardware of the experiment system

Transmission block include a pulsed laser and a half wave plate(Fig 4).

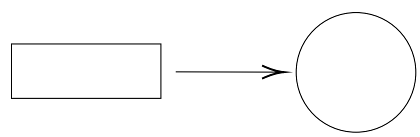


Figure 4. Transmitter Block

Detectors block include two photons detectors, half wave plate and beam splitter(Fig 5).

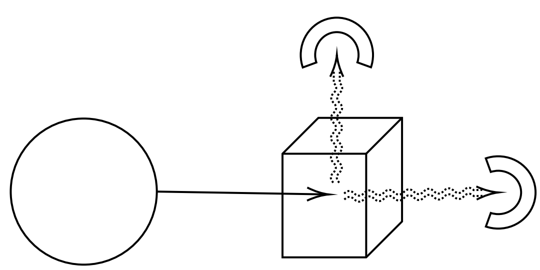


Figure 5. Detector block

A. Communication without Eve

Communication without Eve is a transmittion of data from Alice and Bob detecting the data(Fig 6).

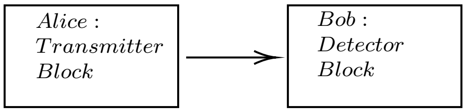


Figure 6. Alice and Bob Block diagram

B. Communication with Eve

Eve Detecting Alice data and transmit data to Bob, Eve is detecting and transmitting block. Eve placed between Alice and Bob(Fig 7).

Figure 7. Alice, Bob and Eve Block diagram

C. differences between the experimental model and the theoretical model

Since sending a single photon at a time requires very expensive equipment, the experiment was done with classical light sent by a laser. the system is identical to the theoretical model except the change of the single photon source to a laser and because new when the wrong base is chosen the beam splitter splits the light to two different beams instead of choosing one path randomly like a single photon. Because we want to illustrate the case of a single photon when both detectors are activated a device will randomly choose a result.

III. RESULTS

A. Alice and Bob communication

the Experiment of Alice and Bob communication was done 3 times, once with 18 bits, once with 50 and once with 100. in every Experiment we calculated the number of times that Alice and Bob chose the same base. because the probability that Alice and Bob chose the same base is 0.5, the experiment's probability distribution can be described by a Binomial distribution, so the expected value for Alice and Bob to chose the same base in a N-bits message E(N) is:

$$E(N) = \frac{N}{2} \pm \frac{\sqrt{N}}{2} \tag{8}$$

the results of Alice and Bob communication for 18 bits are represented in table III and the results of all 3 messages are summed up in table IV

Table III. the results of a message of 18 bits

alice base	alice msg	bob's base	bob's result
x	1	x	1
x	1	+	0
+	0	+	0
x	1	+	1
+	0	+	0
x	1	x	1
+	0	+	1
+	0	x	0
x	1	+	1
+	0	x	1
x	1	+	1
+	0	+	0
+	0	+	1
+	0	x	1
x	1	x	0
+	0	+	0
+	0	x	1
x	1	+	1

Table IV. alice and bob communication summary

Message length	Expected value	Disagriment percantage	Disagriment count	sigma	N sigma
18	9	50%	9	2.1	0
50	25	64%	32	3.5	2.0
100	50	67%	67	5.0	3.4
summary 18+50+100	84	64%	108	6.5	3.7

B. Alice, Bob and Eve communication

the experiment of Alice, Bob and eve communication was done 3 times for the same messages length of the previous experiment. in every simulation we calculated the number of times that Alice and Bob chose the same base and eve was detected. because the probability that Alice and Bob chose the same base is 0.5 and the probability of detecting eve is 0.25, the detection of eve's probability distribution can be described by a Binomial distribution, so the expected value of detecting eve in an N-bits message $E_{eve}(N)$ is:

$$E_{eve}(N) = \frac{N}{4} \pm \frac{\sqrt{3N}}{4} \quad (9)$$

the 18-bits full results are represented in table V where the bits with matched bases but eve was not detected are marked with yellow and the bits that detect eve are marked with red. a summary of all 3 measurements is represented in table IV with the simulation results.

Table V. the results of a message of 18 bits with eve

eve key	bob key	alice key	alice msg	eve msg	bob msg
+	x	+	0	0	1
x	+	+	1	1	0
x	+	x	1	1	0
x	x	x	1	1	1
x	+	x	1	1	1
x	+	+	1	0	0
+	+	x	1	1	1
x	+	x	0	0	1
x	x	x	1	1	1
+	x	+	1	1	0
+	+	x	0	1	1
x	+	x	1	1	1
+	x	+	1	1	1
x	+	x	0	0	1
x	x	x	1	1	1
x	x	+	0	0	0
x	x	x	0	0	0
+	x	x	0	0	1

C. Simulation

Based on a BB84 protocol simulation, a MATLAB GUI as been made. the simulator GUI provide a information that not accessible to a "end user" that communicate via the GUI in real BB84 communication. to distinguish between the information that available we define for this section a "user" as a "end user" that will use the technology and will NOT be accessible to some of the information.
simulator Inputs is:

- Eve present [1 (for true), o (for false)] - real communication users CANT chose if Eve is present. this Input is to simulate the present of Eve, the Test results is the only indicator for the present of Eve for a user.
- Alice massage ["string"]
- Test string size [integer #] - a figure that represent the probabilities to detect Eve as a function of the Test string size is part of the GUI design, such that a end user can choose without deep knowledge of the protocol what is a "safe" enough Test string size.

The output is :

- Bob massage received ["string"] - if the protocol didn't detect Eve the message received successfully if Eve is not present.
if the protocol didn't detect Eve and Eve is present, from a bad choose of the Test string size, Bob will receive a "gibberish" message.
if the protocol detect Eve present The message discard and a indicator string for that will be Bob received message.
- Number of Bits [integer #] - The number of bits used are as follow - $Alice\ message\ length * 7(Matlab\ default\ Binary\ array\ for\ a\ letter) * 1000(for\ a\ large\ margin\ for\ test\ string\ size)$
- Number of match bases of Alice and Bob [integer #]
- Number of match bits from the times that the bases are match [integer #] - the number that indicate if Eve present, this number is NOT the test its purpose its academic and not part of the information that a User accessible to.
- Error [%] - the percentage of mismatches in bits from the times that the bases match. also for academic use, the user is not accessible for this information.
- Test result - if the test not finds Eve a indicator message with the confidence percentage % calculate from the probability to detect Eve with the test string size that the user chosen
if the test finds Eve a string indicate Eve present
if the test not finds Eve but Eve present, the test will indicate that Eve is not present.
This is the information that available to the user, if the test string size is not large enough there is a possibility that Eve will NOT discover.

simulation result figures (8,9,10) show a simple GUI app that deliver the necessary information to a user that need only the knowledge about the test string size, figure 3 are built in the app, for help in this choice. the statistics of the confidence level are a Tool for the user, the rest are for academic purpose.

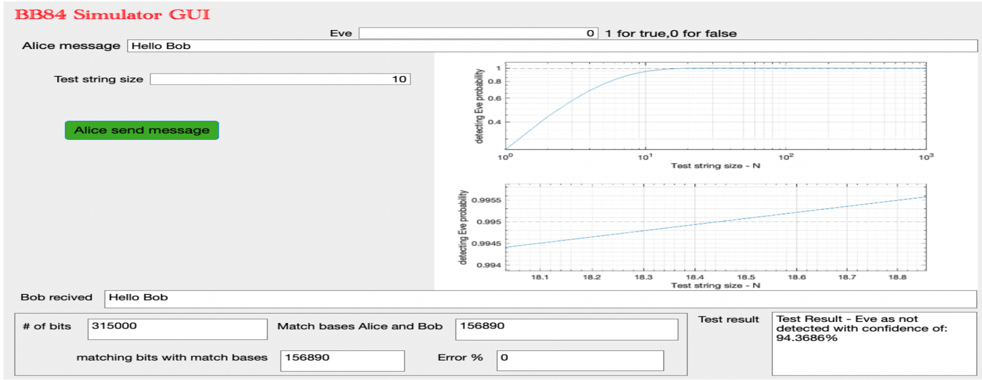


Figure 8. A communication between Alice and Bob when Eve is not present- Alice sends "Hello Bob", the size of the Test string as chosen to be 10. the number of bits/bases is 31500. The number of matched bases between Alice and Bob is 157381 and also the matching bits from the matching bases, from that the error is 0 that correspond to the chose of Eve not present. the test result output indicate that Eve is not present with a Laval of confidence of 94.3686%. Bob received the correct massage with high confidence "Hello Bob".

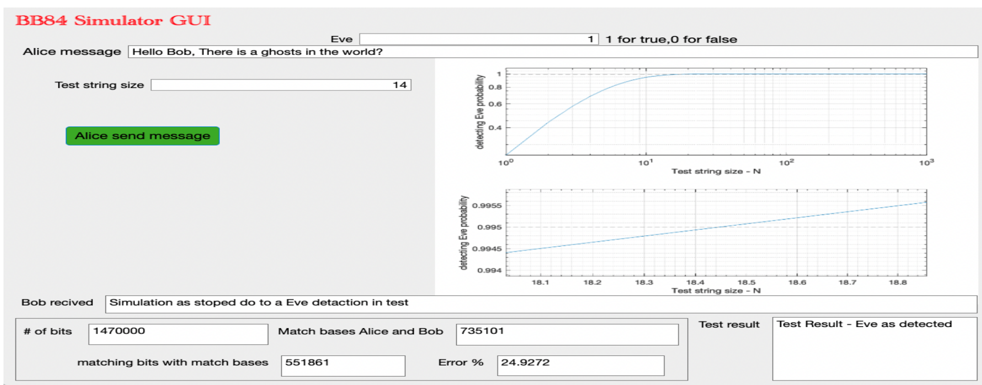


Figure 9. A communication between Alice and Bob when Eve is present- Alice sends "Hello Bob, There is a ghosts in the world?", the size of the Test string as chosen to be 14. the number of bits/bases is 1470000. The number of matched bases between Alice and Bob is 735101 and matching bits from the matching bases is 551851, from that the error is 24.9272 that correspond to the chose of Eve IS present. the test result output indicate that Eve is present. Bob received the indicator massage that a Eve is present, now Bob can say to Alice that something is standing between them in the line of communication, and Alice now know that there is a ghosts in the world

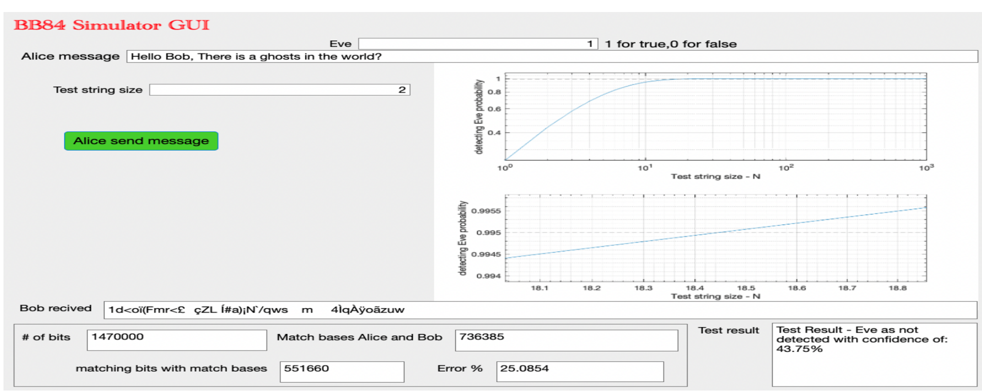


Figure 10. A communication between Alice and Bob when Eve is present- Alice sends "Hello Bob, There is a ghosts in the world?", the size of the Test string as chosen to be 2. the number of bits/bases is 1470000. The number of matched bases between Alice and Bob is 736385 and matching bits from the matching bases is 551660, from that the error is 25.0854 that correspond to the chose of Eve IS present. the test result output indicate that Eve is NOT present with confidence of 43.75% . Bob received a "gibberish" message.

Table VI. Lab results with Eve-Summery

Message length	expected matches	Expected detections value	detection percantage	num of detections	num of matches	sigma	N sigma
18	9	4.5	11%	2	6	3.2	0.8
50	25	12.5	14%	7	30	5.3	1.0
100	50	25	18%	18	43	7.5	0.9
summary 18+50+100	84	42	16%	27	79	9.7	1.5

Table VII. Simulation results with/without Eve-Summery

Message length	Test size	Match Bases	Match Bases detected	Error %	Confidence level in test%	Eve present	Eve detected	Bob received correct message
315000	10	156980	156980	0	94.4	✗	✗	✓
1470000	14	735101	551869	24.9	$\rightarrow$ 100	✓	✓	✓
1470000	2	736385	551660	25.1	43.75	✓	✗	✗

IV. CONCLUSIONS

the protocol was able to detect eve for chose of string size that statistic "big enough", in the case of the bad communication, case of row 3 in table (VII) the test string size chose to 2 with confidence level of 43.75 presence where correspond to a bad communication more than half of the encryption procedures, this show the importance of choosing the test size correctly, one can say that choose it as big as you want, but it will cost in computing time in big message. The detection percentage of eve was a little smaller then expected in the lab, this is because the random device had a tendency to choose one result over the other one (not random), fact that we observe in the laboratory, this may cause by higher sensitively of one detector over the other, or by not ideal angle of transmission that caused by human error. because the ratio of times eve could be detected and was not detected is much less then 50 percent(when Alice and bob choose the same base and a different one from eve) the simulation went as expected without any unexpected results and had about 25 percent of detection of eve and about 50 present of the bases were the same between Alice and bob.

The simple interference GUI show that in the future the BB84 can help every user with no need in understanding the Quantum and mathematics principles behind it.

For now it already shown that BB84 protocol technology is exist, One photons detectors and transmitters are exist, and a experiment made by the Chinese using satellites as been made with successes.

The hardware is Large and expensive, minimize the hardware in the future will give the possibility to communicate safely for every user with a computer and phone, what can be necessary in the era of Quantum computers.

[1] K.Vishi, M.D.Zych, and A.Jøsang, The impact of quantum computing on present cryptography, arXiv preprint arXiv:1804.00200 (2018).

[2] C. G. S.Breidbart and S.Wiesner, Quantum cryptography, or unforgeable subway tokens., Advances in Cryptology , 267 (1983).

[3] M. Planck, On the law of distribution of energy in the normal spectrum, Annalen der physik **4**, 553 (1901).

[4] A. Einstein, On the electrodynamics of moving bodies, Annalen der physik **17(10)**, 891 (1905).

[5] G. N. Lewis, The conservation of photons, Nature **118(2981)**, 874 (1926).

[6] P. A. M. Dirac, A new notation for quantum mechanics, Mathematical Proceedings of the Cambridge Philosophical Society **35(3)**, 416 (1939).

[7] J.L.Park, The concept of transition in quantum mechanics, Foundations of Physics **1 (1)**, 23–33 (1970).

V. APPENDIX

A. FULL MESURMANTS DATA

https://drive.google.com/drive/folders/1m5r0bjuwAIeUmEaCxibBIPQYE3ppbCax?usp=sharing

B. TECHNOLOGIES

systems for quantum encryption:
http://www.idquantique.com/quantum-safe-crypto/
Random number generators :
http://www.idquantique.com/random-number-generation/
Single Photons transmitter and detectors
https://www.ape-berlin.de/en/quantum-dot-single-photon-generation-source/

C. SIMULATION/GUI MATLAB CODE

```
classdef Sim_final < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure matlab.ui.Figure
        BB84SimulatorGUILabel matlab.ui.control.Label
        fortrue0forfalseLabel matlab.ui.control.Label
        TestresultTextArea matlab.ui.control.TextArea
        TestresultTextAreaLabel matlab.ui.control.Label
        BobrecivedTextArea matlab.ui.control.TextArea
        BobrecivedTextAreaLabel matlab.ui.control.Label
        EveEditField matlab.ui.control.NumericEditField
        EveEditFieldLabel matlab.ui.control.Label
        AlicemessageEditField matlab.ui.control.EditField
        AlicemessageEditFieldLabel matlab.ui.control.Label
        TeststringsizeEditField matlab.ui.control.NumericEditField
        TeststringsizeEditFieldLabel matlab.ui.control.Label
        Panel matlab.ui.container.Panel
        ErrorTextArea matlab.ui.control.TextArea
        ErrorTextAreaLabel matlab.ui.control.Label
        matchingbitswithmatchbasesTextArea matlab.ui.control.TextArea
        matchingbitswithmatchbasesTextAreaLabel matlab.ui.control.Label
        ofbitsTextArea matlab.ui.control.TextArea
        ofbitsTextAreaLabel matlab.ui.control.Label
        MatchbasesAliceandBobTextArea matlab.ui.control.TextArea
        MatchbasesAliceandBobTextAreaLabel matlab.ui.control.Label
        AlicesendmessageButton matlab.ui.control.StateButton
        Image matlab.ui.control.Image
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Value changed function: AlicemessageEditField
        function AlicemessageEditFieldValueChanged(app, event)
            value = app.AlicemessageEditField.Value;
            message = value;
            message_length = strlength(message);
        end

        % Callback function
        function EveSwitchValueChanged(app, event)
            value = app.EveSwitch.Value;
            Eve_is_here = value;
        end

        % Value changed function: TeststringsizeEditField
        function TeststringsizeEditFieldValueChanged(app, event)
            value = app.TeststringsizeEditField.Value;
            Test_size = value;
        end
```

```
        % Callback function
        function BobrecivedTextAreaValueChanged(app, event)
            value = app.BobrecivedTextArea.Value;
        end

        % Callback function
        function GetthemessagefromAliceButtonValueChanged(app, event)

            answer = 'what your want to display';
            app.BobrecivedTextArea.Value = answer;
        end

        % Value changed function: AlicesendmessageButton
        function AlicesendmessageButtonValueChanged(app, event)
            message = app.AlicemessageEditField.Value;
            message_length = strlength(message);
            Test_size = app.TeststringsizeEditField.Value;
            Eve_is_here = app.EveEditField.Value;% 0 for false , 1 for true

            %Number of initial random bits
            N = message_length*7*5e3
            N_string = 1:1:N;
            app.ofbitsTextArea.Value= string(N);
            %create base for Alice Bob and Eve and bits string for Alice
            %pseudo - random

            Alice_base = randi([0,1],N,1);
            Alice_Bits = randi([0,1],N,1);
            Bob_base = randi([0,1],N,1);
            Eve_base = randi([0,1],N,1);

            if Eve_is_here == 1
                Eve_Bits = zeros(N,1);
                %transmition bitween 1 and 2 (Alice and Bob OR Alice and Eve)
                for i=N_string
                    if Alice_base(i) == Eve_base(i)
                        Eve_Bits(i) = Alice_Bits(i);
                    else
                        Eve_Bits(i) = randi([0,1],1,1);
                    end
                end
                Bob_Bits = zeros(N,1);
                %transmition bitween 1 and 2 (Alice and Bob OR Alice and Eve)
                for i=N_string
                    if Bob_base(i) == Eve_base(i)
                        Bob_Bits(i) = Eve_Bits(i);
                    else
                        Bob_Bits(i) = randi([0,1],1,1);
                    end
                end

            else
                Bob_Bits = zeros(N,1);
                %transmition bitween 1 and 2 (Alice and Bob OR Alice and Eve)
                for i=N_string
                    if Alice_base(i) == Bob_base(i)
                        Bob_Bits(i) = Alice_Bits(i);
                    else
                        Bob_Bits(i) = randi([0,1],1,1);
                    end
                end
            end

            %find matching bases bitween Alice and Bob
            index_equal_base=Alice_base==Bob_base;

            %bits of match base
            Alice_Bits_same_base = Alice_Bits(
```

```
            index_equal_base);

        Bob_Bits_same_base = Bob_Bits(index_equal_base);

        index_eq_bits = Alice_Bits_same_base==Bob_Bits_same_base;

        %precence of match
        sum_eq_bases = sum(index_equal_base)
        sum_eq_bits =sum(index_eq_bits)
        Pre_of_match_bits = (sum_eq_bases...
            -sum_eq_bits)/sum_eq_bases*100

        app.matchingbitswithmatchbasesTextArea.Value = string(sum_eq_bits);
        app.MatchbasesAliceandBobTextArea.Value = string(sum_eq_bases);
        app.ErrorTextArea.Value = string(Pre_of_match_bits);

        %Test
        Test_Alice = Alice_Bits_same_base(1:Test_size);
        Test_Bob = Bob_Bits_same_base(1:Test_size);
        test_result_array = Test_Alice==Test_Bob;
        test_result_bin = Test_size-sum(test_result_array);



        if test_result_bin == 0
            disp('Test Result - Eve as not detected')
            app.TestresultTextArea.Value = 'Test Result - Eve as not detected with confidence of:'+string((1-(3/4)^Test_size)*100)+'%';
            fprintf('Detecting Eve confidance lavel - %s precence',(1-(3/4)^Test_size)*100)
            disp('message is being encryipt with the key')
            binary = reshape(dec2bin(message, 8).'-'0',1,[]);

            if Test_size == 0
                encriped_Alice_messege = xor(transpose(Alice_Bits_same_base(1:length(binary))),binary);

            else
            transpose(Alice_Bits_same_base(Test_size:Test_size + length(binary)-1));
            encriped_Alice_messege = xor(transpose(Alice_Bits_same_base(Test_size:Test_size + length(binary)-1)), binary);
            disp('encrypted message as sent to Bob')
            end
        else
            disp('Test Result - Eve as detected')
            app.TestresultTextArea.Value = 'Test Result - Eve as detected';
            disp('The protocol as stoped')
        end

        %Bob decript Alice message
        if test_result_bin == 0
            L = length(encriped_Alice_messege);
            mas_bin = xor(transpose(Bob_Bits_same_base...
                (Test_size+1:Test_size +L)), encriped_Alice_messege);
            str = char(bin2dec(reshape(char(mas_bin+'0'), 8,[]).'));
            fprintf('Alice message is: %s ', message);
            fprintf('Bob get the message: %s ', str(1:end));
```

```
                app.BobrecivedTextArea.Value = convertCharsToStrings(str);
            else
                app.BobrecivedTextArea.Value = 'Simulation as stoped do to a Eve detaction in test';
            end
        end

        % Value changing function: ofbitsTextArea
        function ofbitsTextAreaValueChanging(app, event)
            changingValue = event.Value;

        end

        % Value changing function: MatchbasesAliceandBobTextArea
        function MatchbasesAliceandBobTextAreaValueChanging(app, event)
            changingValue = event.Value;

        end

        % Value changing function: matchingbitswithmatchbasesTextArea
        function matchingbitswithmatchbasesTextAreaValueChanging(app, event)
            changingValue = event.Value;

        end

        % Value changing function: ErrorTextArea
        function ErrorTextAreaValueChanging(app, event)
            changingValue = event.Value;

        end

        % Value changed function: EveEditField
        function EveEditFieldValueChanged(app, event)
            value = app.EveEditField.Value;

        end

        % Value changing function: BobrecivedTextArea
        function BobrecivedTextAreaValueChanging(app, event)
            changingValue = event.Value;

        end

        % Value changed function: ofbitsTextArea
        function ofbitsTextAreaValueChanged(app, event)
            value = app.ofbitsTextArea.Value;

        end

        % Value changed function: MatchbasesAliceandBobTextArea
        function MatchbasesAliceandBobTextAreaValueChan(app, event)
            value = app.MatchbasesAliceandBobTextArea.Value;

        end

        % Value changed function: matchingbitswithmatchbasesTextArea
        function matchingbitswithmatchbasesTextAreaValue(app, event)
            value = app.matchingbitswithmatchbasesTextArea.Value;

        end

        % Value changed function: ErrorTextArea
        function ErrorTextAreaValueChanged(app, event)
            value = app.ErrorTextArea.Value;

        end

        % Value changed function: TestresultTextArea
        function TestresultTextAreaValueChanged(app, event)
```

```
                value = app.TestresultTextArea.Value;

            end
        end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Get the file path for locating images
            pathToMLAPP = fileparts(mfilename('fullpath'));

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 983 685];
            app.UIFigure.Name = 'MATLAB App';

            % Create Image
            app.Image = uiimage(app.UIFigure);
            app.Image.Position = [340 121 634 490];
            app.Image.ImageSource = fullfile(pathToMLAPP, 'test help.png');

            % Create AlicesendmessageButton
            app.AlicesendmessageButton = uibutton(app.UIFigure, 'state');
            app.AlicesendmessageButton.ValueChangedFcn = createCallbackFcn(app, @AlicesendmessageButtonValueChanged, true);
            app.AlicesendmessageButton.Text = 'Alice send message';
            app.AlicesendmessageButton.BackgroundColor = [0.3922 0.8314 0.0745];
            app.AlicesendmessageButton.FontSize = 18;
            app.AlicesendmessageButton.Position = [74 376 183 36];

            % Create Panel
            app.Panel = uipanel(app.UIFigure);
            app.Panel.Position = [17 11 668 101];

            % Create MatchbasesAliceandBobTextAreaLabel
            app.MatchbasesAliceandBobTextAreaLabel = uilabel(app.Panel);
            app.MatchbasesAliceandBobTextAreaLabel.HorizontalAlignment = 'right';
            app.MatchbasesAliceandBobTextAreaLabel.FontSize = 16;
            app.MatchbasesAliceandBobTextAreaLabel.Position = [257 66 203 22];
            app.MatchbasesAliceandBobTextAreaLabel.Text = 'Match bases Alice and Bob';

            % Create MatchbasesAliceandBobTextArea
            app.MatchbasesAliceandBobTextArea = uitextarea(app.Panel);
            app.MatchbasesAliceandBobTextArea.ValueChangedFcn = createCallbackFcn(app, @MatchbasesAliceandBobTextAreaValue, true);
            app.MatchbasesAliceandBobTextArea.ValueChangingFcn = createCallbackFcn(app, @MatchbasesAliceandBobTextAreaValu, true);
            app.MatchbasesAliceandBobTextArea.FontSize = 16;
            app.MatchbasesAliceandBobTextArea.Position = [475 64 182 26];

            % Create ofbitsTextAreaLabel
            app.ofbitsTextAreaLabel = uilabel(app.Panel);
```

```
            app.ofbitsTextAreaLabel.HorizontalAlignment = 'right';
            app.ofbitsTextAreaLabel.FontSize = 16;
            app.ofbitsTextAreaLabel.Position = [4 66 68 22];
            app.ofbitsTextAreaLabel.Text = '# of bits ';

            % Create ofbitsTextArea
            app.ofbitsTextArea = uitextarea(app.Panel);
            app.ofbitsTextArea.ValueChangedFcn = createCallbackFcn(app, @ofbitsTextAreaValueChanged, true);
            app.ofbitsTextArea.ValueChangingFcn = createCallbackFcn(app, @ofbitsTextAreaValueChanging, true);
            app.ofbitsTextArea.FontSize = 16;
            app.ofbitsTextArea.Position = [87 65 165 25];

            % Create matchingbitswithmatchbasesTextAreaLabel
            app.matchingbitswithmatchbasesTextAreaLabel = uilabel(app.Panel);
            app.matchingbitswithmatchbasesTextAreaLabel.HorizontalAlignment = 'right';
            app.matchingbitswithmatchbasesTextAreaLabel.FontSize = 16;
            app.matchingbitswithmatchbasesTextAreaLabel.Position = [13 21 239 22];
            app.matchingbitswithmatchbasesTextAreaLabel.Text = ' matching bits with match bases';

            % Create matchingbitswithmatchbasesTextArea
            app.matchingbitswithmatchbasesTextArea = uitextarea(app.Panel);
            app.matchingbitswithmatchbasesTextArea.ValueChangedFcn = createCallbackFcn(app, @matchingbitswithmatchbasesTextArea
            , true);
            app.matchingbitswithmatchbasesTextArea.ValueChangingFcn = createCallbackFcn(app, @matchingbitswithmatchbasesTextArea
            , true);
            app.matchingbitswithmatchbasesTextArea.FontSize = 16;
            app.matchingbitswithmatchbasesTextArea.Position = [267 19 114 26];

            % Create ErrorTextAreaLabel
            app.ErrorTextAreaLabel = uilabel(app.Panel);
            app.ErrorTextAreaLabel.HorizontalAlignment = 'right';
            app.ErrorTextAreaLabel.FontSize = 16;
            app.ErrorTextAreaLabel.Position = [414 21 61 22];
            app.ErrorTextAreaLabel.Text = 'Error %';

            % Create ErrorTextArea
            app.ErrorTextArea = uitextarea(app.Panel);
            app.ErrorTextArea.ValueChangedFcn = createCallbackFcn(app, @ErrorTextAreaValueChanged, true);
            app.ErrorTextArea.ValueChangingFcn = createCallbackFcn(app, @ErrorTextAreaValueChanging, true);
            app.ErrorTextArea.FontSize = 16;
            app.ErrorTextArea.Position = [490 20 150 25];

            % Create TeststringsizeEditFieldLabel
            app.TeststringsizeEditFieldLabel = uilabel(app.UIFigure);
            app.TeststringsizeEditFieldLabel
```

```
            .HorizontalAlignment = 'right';
            app.TeststringsizeEditFieldLabel.FontSize = 16;
            app.TeststringsizeEditFieldLabel.Position = [24 575 111 22];
            app.TeststringsizeEditFieldLabel.Text = 'Test string size';

            % Create TeststringsizeEditField
            app.TeststringsizeEditField = uieditfield(app.UIFigure, 'numeric');
            app.TeststringsizeEditField.ValueChangedFcn = createCallbackFcn(app, @TeststringsizeEditFieldValueChanged, true);
            app.TeststringsizeEditField.FontSize = 16;
            app.TeststringsizeEditField.Position = [143 575 122 22];

            % Create AlicemessageEditFieldLabel
            app.AlicemessageEditFieldLabel = uilabel(app.UIFigure);
            app.AlicemessageEditFieldLabel.HorizontalAlignment = 'right';
            app.AlicemessageEditFieldLabel.FontSize = 18;
            app.AlicemessageEditFieldLabel.Position = [21 611 122 23];
            app.AlicemessageEditFieldLabel.Text = 'Alice message';

            % Create AlicemessageEditField
            app.AlicemessageEditField = uieditfield(app.UIFigure, 'text');
            app.AlicemessageEditField.ValueChangedFcn = createCallbackFcn(app, @AlicemessageEditFieldValueChanged, true);
            app.AlicemessageEditField.FontSize = 16;
            app.AlicemessageEditField.Position = [151 610 823 24];

            % Create EveEditFieldLabel
            app.EveEditFieldLabel = uilabel(app.UIFigure);
            app.EveEditFieldLabel.HorizontalAlignment = 'right';
            app.EveEditFieldLabel.FontSize = 16;
            app.EveEditFieldLabel.Position = [348 645 70 23];
            app.EveEditFieldLabel.Text = 'Eve';

            % Create EveEditField
            app.EveEditField = uieditfield(app.UIFigure, 'numeric');
            app.EveEditField.ValueChangedFcn = createCallbackFcn(app, @EveEditFieldValueChanged, true);
            app.EveEditField.FontSize = 16;
            app.EveEditField.Position = [426 646 97 22];

            % Create BobrecivedTextAreaLabel
            app.BobrecivedTextAreaLabel = uilabel(app.UIFigure);
            app.BobrecivedTextAreaLabel.HorizontalAlignment = 'right';
            app.BobrecivedTextAreaLabel.FontSize = 16;
            app.BobrecivedTextAreaLabel.Position = [17 123 91 22];
            app.BobrecivedTextAreaLabel.Text = 'Bob recived';

            % Create BobrecivedTextArea
            app.BobrecivedTextArea = uitextarea(app.UIFigure);
            app.BobrecivedTextArea.ValueChangingFcn = createCallbackFcn(app,
```

```
            @BobrecivedTextAreaValueChanging, true);
            app.BobrecivedTextArea.FontSize = 16;
            app.BobrecivedTextArea.Position = [122 121 849 26];

            % Create TestresultTextAreaLabel
            app.TestresultTextAreaLabel = uilabel(app.UIFigure);
            app.TestresultTextAreaLabel.HorizontalAlignment = 'right';
            app.TestresultTextAreaLabel.FontSize = 16;
            app.TestresultTextAreaLabel.Position = [690 88 78 22];
            app.TestresultTextAreaLabel.Text = 'Test result';

            % Create TestresultTextArea
            app.TestresultTextArea = uitextarea(app.UIFigure);
            app.TestresultTextArea.ValueChangedFcn = createCallbackFcn(app, @TestresultTextAreaValueChanged, true);
            app.TestresultTextArea.FontSize = 16;
            app.TestresultTextArea.Position = [783 11 188 101];

            % Create fortrue0forfalseLabel
            app.fortrue0forfalseLabel = uilabel(app.UIFigure);
            app.fortrue0forfalseLabel.FontSize = 18;
            app.fortrue0forfalseLabel.Position = [530 645 163 23];
            app.fortrue0forfalseLabel.Text = '1 for true,0 for false';

            % Create BB84SimulatorGUILabel
            app.BB84SimulatorGUILabel = uilabel(app.UIFigure);
            app.BB84SimulatorGUILabel.FontName = 'Academy Engraved LET';
            app.BB84SimulatorGUILabel.FontSize = 26;
            app.BB84SimulatorGUILabel.FontWeight = 'bold';
            app.BB84SimulatorGUILabel.FontColor = [1 0 0];
            app.BB84SimulatorGUILabel.Position = [17 633 240 35];
            app.BB84SimulatorGUILabel.Text = 'BB84 Simulator GUI';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = Sim_final

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```