THESIS APPROVAL

GRADUATE SCHOOL, KASETSART UNIVERSITY

Master of Science (Physics)

DEGREE

Physics	Physics
FIELD	**DEPARTMENT**

TITLE: Experimental Studies of Quantum Cryptography in Optical Fiber

NAME: Mr. Santhad Phithakwongsaphorn

THIS THESIS HAS BEEN ACCEPTED BY

THESIS ADVISOR

(Assistant Professor Surasak Chiangga, Dr.rer.nat.)

THESIS CO-ADVISOR

(Assistant Professor Cherdsak Kunsombat, Ph.D.)

THESIS CO-ADVISOR

(Associate Professor Utsanee Leerawat, Ph.D.)

DEPARTMENT HEAD

(Associate Professor Siwaporn Sahavat, M.Sc.)

APPROVED BY THE GRADUATE SCHOOL ON .....

DEAN

(Associate Professor Gunjana Theeragool, D.Agr.)

THESIS

EXPERIMENTAL STUDIES OF QUANTUM CRYPTOGRAPHY IN OPTICAL FIBER

SANTHAD PHITHAKWONGSAPHORN

A Thesis Submitted in Partial Fulfillment of
the Requirements for the Degree of
Master of Science (Physics)
Graduate School, Kasetsart University
2009

Santhad Phithakwongsaphorn 2009: Experimental Studies of Quantum Cryptography in Optical Fiber. Master of Science (Physics), Major Field: Physics, Department of Physics. Thesis Advisor: Assistant Professor Surasak Chiangga, Dr.rer.nat. 64 pages.

A quantum key distribution (QKD) system can create a shared secret cryptographic key over an unsecured optical link. These systems use the fundamental quantum properties of single photons to guarantee the security of the shared key, which is commonly called the net key. The net keys generated in this manner, and at sufficiently high rates, make use of a one-time-pad cipher for encryption of broadband communications links. A number of groups have developed experimental QKD systems operating in both free-space and optical fiber.

The goal of this thesis was to design and construct the optical fiber quantum cryptographic system based on the B92 protocol over the standard telecommunications fiber. The transmitter employed two 850-nm wavelength vertical-cavity surface-emitting lasers, which were directly modulated its injection current by applying a train of 2 ns electrical pulses at a repetition rate of 10 KHz. The output laser pulses were reduced the intensity to a mean photon number of 0.5 photons per pulse. Each sequence of pulses was then assigned by the passive optics at the transmitter to one of two polarization states: horizontal; H, and right circular; R. Our receiver contained a fiber polarization controller to recover the photon's polarization state, and two avalanche photodiodes operating in Geiger mode were used to detect single photons. We observed the visibility of the transmitter for the H and R which were 0.97 and 0.99 respectively. The visibility of the receiver for the H and R which were 0.38 and 0.69 respectively. These visibilities indicated that the quantum-bit errors were approximately 50.39%.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

ACKNOWLEDGEMENTS

I would like to grateful thank and deeply indebted to my advisor and teacher, Asst. Prof. Dr.Surasak Chiangga, who has offered me with knowledge, understanding, skill, encouragement and valuable suggestion for completely writing of thesis.

I would like to thank my thesis committees, Asst. Prof. Dr. Cherdsak Kunsombat, Assoc. Prof. Dr. Nason Phonpoke, Assoc. Prof. Dr. Utsanee Leerawat and Dr. Sutee Boonchuay for their suggestions and corrections to my thesis.

I sincerely thank Mr. Watchara Pornkaveerat, who has supported in my thesis about laser driving electronics and coincident electronic circuit.

Finally, the advantage of this thesis, I am especially appreciated my parents and my brothers for their continuing encouragements.

Santhad Phithakwongsaphorn

January 2009

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

cps	=	count per second
cm	=	centimeter
Km	=	Kilometer
KHz	=	Kilohertz
Kbits/s	=	Kilobits per second
mm	=	millimeter
nm	=	nanometer
ns	=	nanosecond
s	=	second
Mbits/s	=	Megabits per second

EXPERIMENTAL STUDIES OF QUANTUM CRYPTOGRAPHY IN OPTICAL FIBER

INTRODUCTION

Quantum cryptography or quantum key distribution (QKD), first proposed by Bennett and Brassard in 1984 (Bennett and Brassard, 1984) is a means of distributing a verifiably secure key, between two or more users, over an unsecured channel. Uniquely, the QKD system provides a method for distributing this type of key in a verifiably secure manner. A secure method of encryption-the "one-time pad" approach was proposed in 1917 by Vernam, and was proven absolutely secure by Shannon in 1949 (Shannon, 1949). This encryption technique relies on a key that is truly random, is as long as the message itself, and is used only once. Bennett and Brassard's original QKD protocol (BB84) employs two incompatible pairs of conjugate quantum observables, for example, circular and linear-polarization states, to encode the data. In modified Bennett protocol (B92), two nonorthogonal states are utilized, for example, two nonorthogonal linear-polarization states (Bennett, 1992).

To date, A number of groups have developed experimental QKD systems operating in both free-space (Rarity *et al*., 2001; Bienfang *et al*., 2004) and optical fiber (Bethune *et al*., 2002; Elliott *et al*., 2003). The first study of a fiber-based polarization coding QKD system with silicon detectors was reported in 1994 (Breguet *et al*., 1994). Townsend (1998) and Gordon *et al*. (2004) reported similar systems in the 800 nm wavelength region using standard single-mode fiber. Tang *et al*. (2006) reported a high speed polarization coding QKD system operating at a sifted-key rate over Mbits/s. Lodewyck *et al*. (2007) reported a QKD over 25 Km with an all-fiber continuous-variable system. More recently, Pirandola *et al*. (2008) reported a characterization of collective Gaussian attacks and security of coherent-state quantum cryptography by analyzing the asymptotic secret-key rates which are achievable with coherent states, joint measurements of the quadratures and one way classical communication.

There are two main general problems with sending photons over a free-space link: transmission losses and background light. Since the signal is not transmitted in a guiding medium (such as a fiber optic cable) the energy can spread out leading to transmission losses. Additionally, extraneous background light can also couple into the receiver telescope leading to more background noise and an increased error rate. The errors induced by the background light can be reduced to a reasonable level by using spectral filtering, spatial filtering, and temporal discrimination with a coincidence window of a few nanoseconds (Gisin *et al*., 2002). For the transmission losses, there are a number of effects due to the atmosphere which play a role in the transmission efficiency of the free-space link. The atmosphere itself has particular transmission efficiency for light due to atmospheric extinction of the photons as they travel through the air. Atmospheric extinction refers to the process of photons interacting with air molecules, aerosol particles, and water droplets through scattering and absorption. These processes lead to the loss of some photons and an overall extinction of the light (Lindental, 2006). Fortunately, the atmosphere has a high transmission window (~85%) at a wavelength of about 800 nm where commercial, high-efficiency photon detection modules exist. The fiber loss at 850-nm wavelength (~2.2 $dBKm^{-1}$) is too high to achieve transmission distances of >50 Km. However, for short distance (~10 Km) applications of QKD in networks short wavelength systems are likely to offer the highest key exchange rate due to high-efficiency photon detection modules. Standard telecommunications fiber is not single-mode at a wavelength of 850 nm and this might be thought to prevent the implementation of QKD over deployed fiber networks.

There are many different protocols for quantum key distribution, a good overview of several QKD schemes can be found in a review paper by Gisin *et al*. Quantum cryptography with the BB84 protocol can be performed ideally with single photons (Beveratos *et al*., 2002) or, more practically, with weak coherent laser pulses (Bennett *et al*., 1992) However, the weak coherent laser pulse schemes are open to the photon number splitting attack since more than one photon is sometimes created in a pulse. Eve could then split off one photon for her to measure from each multi-pair event and gain information about the key. A method for overcoming the photon-

number-splitting attack for the weak laser pulse implementations has been developed using decoy states (Hwang, 2003). Quantum key distribution protocols have also been extended to use entangled qubit pairs as in the Ekert91 protocol proposed by Ekert in 1991 (Ekert, 1991) or the BBM92 protocol by Bennett, Brassard, and Mermin in 1992 (Bennett *et al*., 1992).

Now, quantum cryptography has come out from laboratory to real products, but a numbers of practical problems remain to be solves. The significant drawbacks of many practical quantum cryptography systems are unavailable single-photon source and imperfect detectors. The most of practical sources rely on attenuating laser pulses. One disadvantage of these sources is the pulse contains more than one photon with significant probability. Eavesdropper can harm those systems through a beam splitter attack. Less efficient single-photon detectors have obviously impacted on the bit rate and maximum span length (Gisin *et al*., 2002).

In this thesis, we implemented the B92 protocol. Although it is well known that the B92 protocol is less secure than the BB84 protocol, it is widely used in the laboratory study of the physical-layer limitations of a QKD system, such as timing jitter, dead time, and polarization leakage. By adding two additional APDs and faint laser sources, a B92 QKD test-bed could be converted to BB84.

We have been developing QKD to improve some of the implementation features in quantum cryptography, including hardware design, software integration and rate of key generation. Our QKD system described here has been developed from implementing the BB84 protocol by Deachapunya (2002) and Panthong (2005). Moreover, our QKD system is similar to Tang *et al*. (2006) by using the VCSELs and high detection efficiency of Si-APDs, but different at the sifted-key rates.

OBJECTIVES

The objectives of this thesis are as follows,

1. To study the B92 protocol and encode/decode the polarization states of single photons based on it.

2. To design and construct the quantum cryptography system based on B92 protocol.

3. To develop the computer program for the quantum cryptographic system.

4. To test the ability of sending and receiving the encoded single photons over an optical fiber.

LITERATURE REVIEW

Qubit

In classical computing, the main fundamental computational unit is the bit. It can either have the value of 0 or 1, such as switch on-off, or TTL signal, etc (shown in figure 1). The analogous concept for quantum computing is the quantum bit or qubit. A qubit can be in one of two orthogonal states $|0\rangle$ or $|1\rangle$, or any coherent superposition of these two states $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Any two level quantum systems can be used to encode a qubit, such as, spins of electrons or electron in hydrogen atom, or the polarization of photons (shown in figure 1).

All of the experiments in this work will use the polarization of photons as qubits with the two orthogonal states being $|H\rangle$ and $|V\rangle$ where H and V refer to the horizontal and vertical polarizations of a photon with respect to a suitable frame of reference. The $|H\rangle$ and $|V\rangle$ states correspond to the computational basis states $|0\rangle$ and $|1\rangle$ respectively. Photons are ideal qubits for quantum communication and cryptography schemes because of their weak interaction with each other and most matter. This weak interaction translates into low decoherence rates, so that the qubits maintain their quantum states for a long time. Also, they move at the speed of light which makes it possible to transmit them very quickly over large distances.

Classical Cryptography

The usual situation is this: Party A (usually called Alice) wants to send a message to party B (named Bob) in a secure way. An eavesdropper (Eve) who gets hold of the message should not be able to gain any information about its contents.

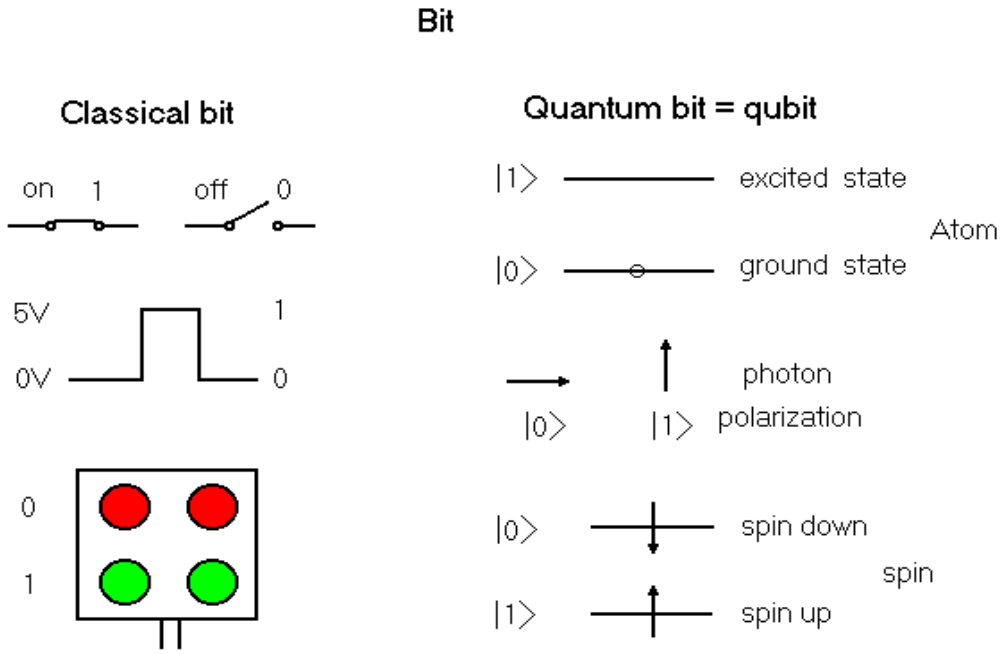

Figure 1 The classical bits versus the quantum bits.

Source: Deachapunya (2002)**.**

As a first step in the mathematical analysis of cryptography, it is necessary to idealize the situation suitably, and to define in a mathematically acceptable way what we shall mean by a secrecy system. A “schematic” diagram of a general secrecy system (shown in Figure 2). At the transmitting end (usually called Alice) there are two information sources—a message source and a key source. The key source produces a particular key from among those which are possible in the system. This key is transmitted by some means over secure channel, supposedly not interceptible, for example by messenger, to the receiving end (called Bob). The message source produces a message (plain text) which is enciphered and the resulting cryptogram (cipher text) sent to Bob by a possibly interceptible means, for example radio. At Bob end the cipher text and key are combined in the decipherer to recover the plain text.

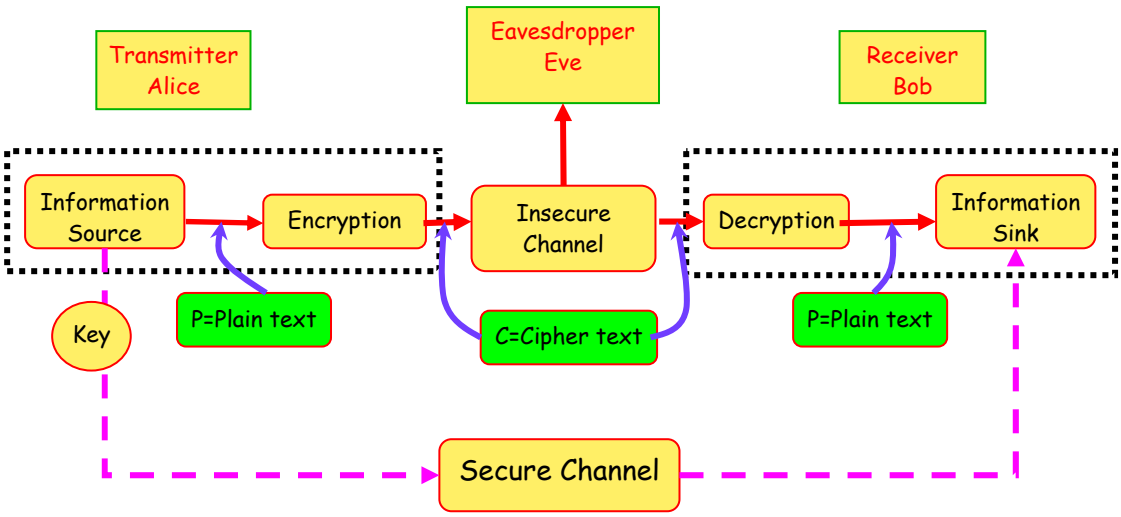

Figure 2 The classical cryptographic communication system.

Source: Lomonaco (2009).

The Vernam Cipher

There is one provably secure cryptographic scheme: the Vernam Cypher or One-Time Pad, which requires a secret random key that Alice and Bob share. Since Quantum Key Distribution will rely on this protocol, it is explained more fully here. The protocol relies on a secret random bit string (the secret key) known only to Alice and Bob. The secret key must be the same length as the data to encrypt. By using the secret key only once to encrypt and decrypt the data it is impossible for anyone who receives only the encrypted data to decrypt it without knowing the secret key. However, if the secret key is used more the once, there are statistical and numerical techniques that an eavesdropper can use to begin to discover the secret key and decipher the data.

A simple implementation of the Vernam cipher is using the bitwise XOR (exclusive-OR) operation of digital logic on the key and data to both encrypt and decrypt the message. An example of encrypting and decrypting a short ASCII

message is given in Table 1. The requirement of only using the key once to encrypt and decrypt can be illustrated with a simple attack on two different messages that were encrypted using the same key as in Table 1. If the two encrypted messages are combined with the XOR operation, the result will be the XOR of the two original messages with the key removed. Since the message bit strings are no longer random without the key, statistical techniques can then be used to rapidly recover the two original messages.

Table 1 The implementation of the Vernam cipher with the XOR operation.

Algorithm	ASCII Message	Message
Message (Alice)	1001000 1101001	" Hi "
Key (Alice)	0010111 0100101	
Encrypted Message (Alice)	1011111 1001100	" -L "
Encoded Message Received (Bob)	1011111 1001100	" -L "
Key (Bob)	0010111 0100101	
Decrypted Message (Bob)	1001000 1101001	" Hi "

Source: Erven (2007**).**

The security of the Vernam cipher relies on the following:

• The key must be random to avoid statistical attacks.

• The key must be securely transported to Alice and Bob so that its secrecy is assured.

Classically, both of these requirements are hard to accomplish securely. For the first requirement, generating truly random numbers is hard, and any patterns in the key can lead to successful statistical attacks. The second requirement is even more difficult, since transported keys can in principle always be intercepted, copied by Eve, and sent on to Alice and Bob without their knowledge. Eve would then be able to listen to any communication between Alice and Bob which used those keys. Quantum

key distribution provides a solution to these problems, allowing one to have a provably secure encryption/decryption protocol.

The B92 Protocol

The so- called B92 protocol was chosen here since it is simpler to implement in practice than the BB84 protocol, as only two states are required instead of four. The polarization encoded version of B92 proceeds as follows for an idealized system. The transmitter “Alice” and the receiver “Bob” each generate an independent random bit sequence. Alice then transmits her random bit sequence to Bob using a clocked sequence of linearly polarized individual photons with polarization angles chosen according to her bit values as given by $0° \equiv 0$ and $45° \equiv 1$. In each time period, Bob makes a polarization measurement on an incoming photon by orientating the transmission axis of his polarizer according to his bit value as given by $-45° \equiv 0$ and $90° \equiv 1$. It can be seen that Bob will only detect a photon (with probability one half) in the time slots where his polarizer is not crossed with that of Alice. We refer to these instances as “unambiguous” since when they occur, Alice and Bob can be sure that their polarization settings were not orthogonal and, consequently, that their bit values were the same (both 0 and both 1). Conversely, the instances in which Bob receives no photon are referred to as “ambiguous” since they can arise either from the cases where Alice’s and Bob’s polarizers were crossed or from the cases where the polarizers were not crossed, but Bob failed (with probability one half) to detect a photon. Bob then uses an authenticated public channel to inform Alice of the time slots in which he obtained an unambiguous result (one in four on average) and they use the shared subset of their initial random bit sequences represented by these time slots as a key. The level of intervention by an eavesdropper “Eve” can then be quantified in the usual way by analyzing the error rate for the key exchange.

To complete the B92 experiment, Bob records randomly (Y or N) to receive bits, and then Bob tell Alice the result of recording received bits. Only bits, recorded to Y, are used to be a secret key. For example is shown in Table 2.

Table 2 The example for the QKD system based on the B92 protocol.

Algorithm	Coding and encoding bit string			
Alice's bits string	1	0	1	0
Sent by Alice	+ 45°	V	+ 45°	V
Bob's result	− 45°	− 45°	H	H
Bob's bits string	0	0	1	1
Bob's recording	N	N	Y	N

Source: Hughes *et al*. (2000)

It is important to note that B92 is not as inherently secure as BB84, since Eve can, in principle, perform a potentially undetectable intercept-resend attack. With this type of attack, Eve chooses to only resend a photon to Bob when she obtains an unambiguous measurement outcome and, hence, knows Alice's polarization setting. This would not cause any depolarization induced errors in the transmission, but would lead to a decreased photon arrival rate that would alert Bob to Eve's presence. How ever, if the quantum channel is lossy (as is the case with optical fiber) then Eve can, in principle, substitute a lower loss channel to compensate for her reduced photon transmission rate and, hence, avoid detection. Various approaches have been discussed to avoid this problem including the use of bright reference pulses in the original interferometric version of B92 (Bennett, 1992). We mention other defenses further below. But also note that the QKD system described here can be developed to implement the BB84 protocol with several straightforward changes. BB84 does not suffer from security deficiency since, with four polarization states, Eve cannot be sure that she has determined the state of any given photon with deterministic certainly.

Quantum Bit Error Rate

The QBER is defined as the ratio of wrong bits to the total number of bits received and is normally on the order of a few percent. We can express it as a function of rates,

$$QBER = \frac{N_{wrong}}{N_{right} + N_{wrong}} = \frac{R_{error}}{R_{sift} + R_{error}} \approx \frac{R_{error}}{R_{sift}} . \quad (1)$$

here the sifted key corresponds to the case in which Alice and Bob made compatible choices of bases, hence its rate is half that the raw key.

The raw rate is essentially the product of the pulse rate f_{rep}, the mean number of photon per pulse μ, the probability t_{link} of a photons arriving at the analyzer, and the probability η of the photon's being detected:

$$R_{sift} = \frac{1}{2} R_{raw} = \frac{1}{2} q f_{rep} \mu t_{link} \eta . \quad (2)$$

The factor q ($q \le 1$, typically 1 or $1/2$) must be introduced for some phase-coding setups in order to correct for noninterfering path combinations.

One can identify three different contributions to R_{error}. The first arises from photons that end up in the wrong detector due to imperfect interference or polarization contrast. The rate R_{opt} is given by the product of the sifted-key rate and the probability p_{opt} of a photon's going to the wrong detector:

$$R_{opt} = R_{sift} p_{opt} = \frac{1}{2} q f_{rep} \mu t_{link} p_{opt} \eta . \quad (3)$$

For a given setup, this contribution can be considered as an intrinsic error rate indicating its suitability for use in QC.

The second contribution, $R_{\det}$, arises from the detector dark counts (or from remaining environmental stray light in free-space setups). This rate is independent of the bit rate. Of course, only dark counts falling within the short time window when a photon is expected give rise to errors,

$$R_{\text{det}} = \frac{1}{2}\frac{1}{2} f_{rep} p_{dark} n , \tag{4}$$

Where p_{dark} is the probability of registering a dark count per time window and per detector, and n is the number of detectors. The two factors of $1/2$ are related to the fact that a dark-count has a 50% chance of happening when Alice and Bob have chosen incompatible bases (and is thus eliminated during sifting) and a 50% chance of occurring in the correct detector.

Finally, error counts can arise from uncorrelated photons due to imperfect photon sources:

$$R_{acc} = \frac{1}{2}\frac{1}{2} p_{acc} f_{rep} t_{link} n \eta . \tag{5}$$

This factor appears only in systems based on entangled photons, where the photons belonging to different pairs but arriving in the same time window are not necessarily in the same state. The quantity p_{acc} is the probability of finding a second pair within the time window, knowing that a first one was created.

The QBER can now be expressed as follows:

$$QBER = \frac{R_{opt} + R_{\text{det}} + R_{acc}}{R_{sift}} \tag{6}$$

$$= p_{opt} + \frac{p_{dark} n}{t_{link} \eta 2 q \mu} + \frac{p_{acc}}{2 q \mu} \tag{7}$$

$$= QBER_{opt} + QBER_{\text{det}} + QBER_{acc} . \tag{8}$$

We now analyze these three contributions. The first one, $QBER_{opt}$, is independent of the transmission distance (it is independent of t_{link}). It can be considered as a measure

of the optical quality of the setup, depending only on the polarization or interference fringe contrast. The technical effort needed to obtain and, more importantly, to maintain a given $QBER_{opt}$ is an important criterion for evaluating different QC setups. In polarization-based systems, it is rather simple to achieve a polarization contrast of 100:1, corresponding to a $QBER_{opt}$ of 1%. In fiber-based QC, the problem is to maintain this value in spite of polarization fluctuations and depolarization in the fiber link. For phase-coding setups, $QBER_{opt}$ and the interference visibility are related by

$$QBER_{opt} = \frac{1-V}{2}. \tag{9}$$

A visibility if 98% thus translates into an optical error rate of 1%. Such a value implies the use of well-aligned and stable interferometers. In bulk optics, perfect mode overlap is difficult to achieve, but the polarization is stable. In single-mode fiber interferometers, on in contrast, perfect mode overlap is automatically achieved, but the polarization must be controlled, and chromatic dispersion can constitute a problem.

The second contribution, $QBER_{\mathrm{det}}$, increase with distance, since the dark-count rate remains constant while the bit rate goes down like t_{link}. It depends entirely on the ratio of the dark-count rate to the quantum efficiency. At present, good single-photon detectors are not commercially available for telecommunications wavelengths. The span of QC is not limited by decoherence. As $QBER_{opt}$ is essentially independent of the fiber length, it is detector noise that limits the transmission distance.

Finally, the $QBER_{acc}$ contribution is present only in some two-photon schemes in which multiphoton pulses are processed in such a way that they do not necessarily encode the same bit value. Although all systems have some probability of multiphoton pulses, in most these contribute only to the information available to Eve

and not to the QBER. However, for implementations featuring passive choice by each photon, the multiphoton pulses do not contribute to Eve's information but only to the error rate.

The fiber link transmission decreases exponentially with length. The fraction of bits lost due to error correction and privacy amplification is a function of QBER and depends on Eve's strategy.

Weak Coherent Pulses

The security of the QKD system is based on the fact that single quantum particles are used to transmit information. Unfortunately, the existing single photon sources are not in a state where it seems practical to use them for quantum cryptography systems which are supposed to be close to an application.

One way to bypass the problem of a missing practical single photon source is the use of weak coherent pulses instead of genuine single photon sources: The numbers of photons n in pulses of a pulsed laser beam is distributed according to Poissonian statistics,

$$p(n) = \frac{\mu^n}{n!} e^{-\mu}, \qquad \text{with} \qquad \mu := \langle n \rangle \quad \text{(mean photon number)} \qquad (10)$$

Here, $p(n)$ denotes the probability of finding n photons in a pulse of a coherent beam described by a mean photon number μ. The form of the Poisson distribution is illustrated in Figure 3 for three values of the mean photon number.

The probability that at least one photon from a pulse with n photons is successfully transmitted is

$$\eta_n = 1 - (1 - \eta)^n \qquad (11)$$

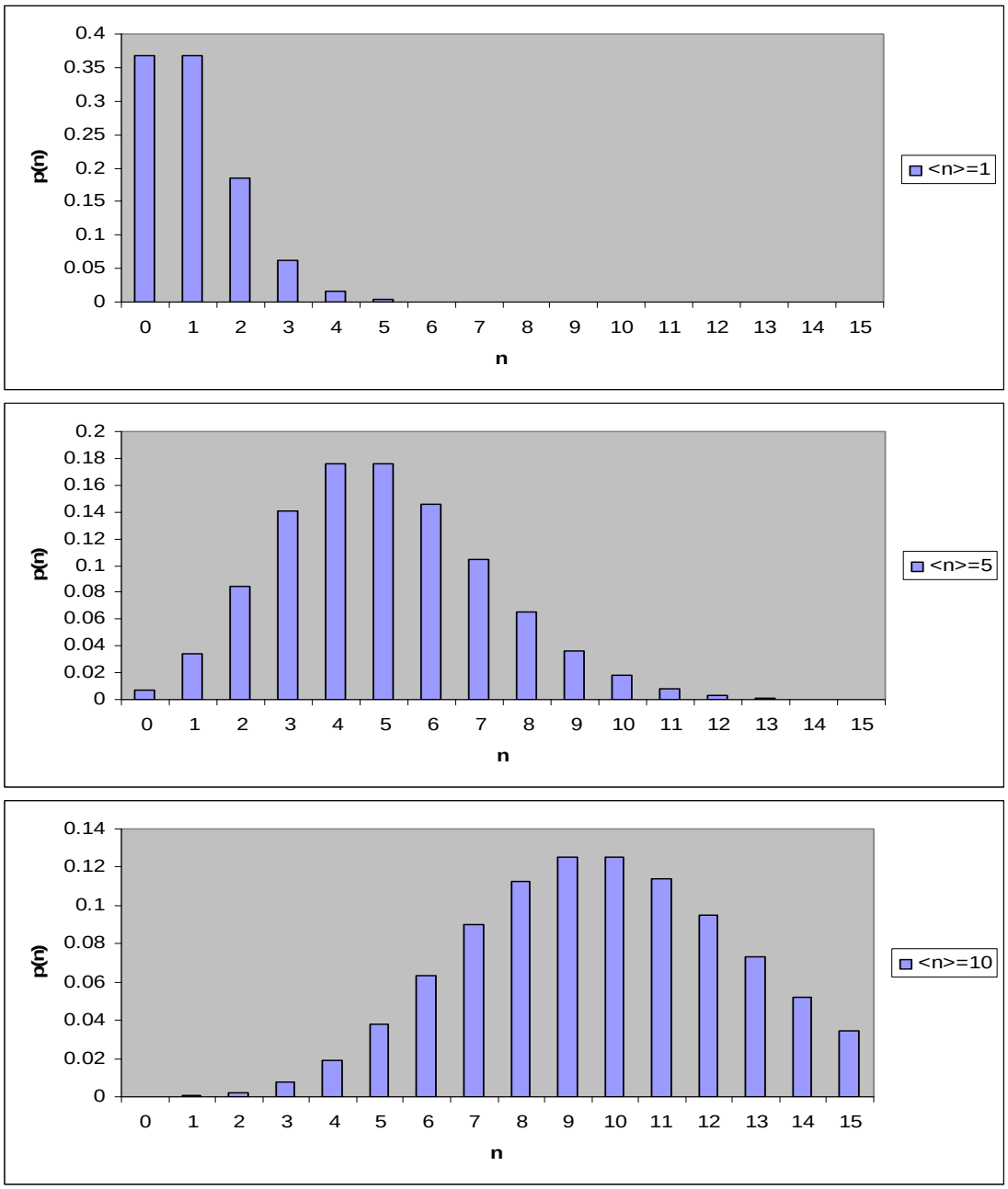

Figure 3 The Poisson form of the photon-count distribution for light beams of constant intensity.

MATERIALS AND METHODS

Materials

1. Metallic neutral density filter: Model FBR-ND01, Newport
2. Fiber polarization controller: Model F-POL-PC,Newport
3. Single mode VCSEL, $\lambda = 850$ nm, P = 30 mW: Model HFE4093-332, Finisar
4. Fiber optic collimator: Model F-C5-F2-780, Newport
5. Polarizing cube beamsplitter: Model 03PS062, Melles Griot
6. Broadband polarizing cube beamsplitter: Model 05FC16PB.5, Newport
7. Quarter-wave plate: Model 02WRQ007, Melles Griot
8. Multimode Coupler: Model F-CPL-M22855, Newport
9. Single-mode fiber cable: Model UFC9201, Interlink
10. Multimode fiber cable: Model UFC6202, Interlink
11. Single Photon Counting Module Array: Model SPCM-AQ4C, PerkinElmer
12. Interface board for SPCM-AQ4C: Model SPCM-AQ4C-IO, PerkinElmer
13. Electronic devices
14. Computer and accessory devices
15. Interface Board: Model ET-PCI8255 V3, ETT
16. Interface Board: Model NI PCI-6221, National Instruments
17. Photodiode: Model DET210, Thorlabs

Methods

The Transmitter Section

The transmitter uses two nonorthogonal polarization states of the single photons to implement the B92 protocol.

1. Electronic part

The laser driver electronics of the transmitter module (Pornkaveerat, 2008) was controlled by the PCI card (ET-PCI8255 V3) for addressing randomly the vertical-cavity surface-emitting lasers (VCSELs).

2. Optical part

The optical alignments of the passive optic of the transmitter module (Alice) have been designed and constructed for set the photon polarization states to $|H\rangle$ and $|R\rangle$ depending on whether the binary number is a “0” or “1” respectively(shown in figure 4). Each VCSEL was directly modulated its injection current by applying a train of 2 ns electrical pulses at a repetition rate of 10 KHz. The beams are then polarized horizontally by the 5 mm cube polarizing beam splitters (PBS). A quarter-wave plate (QP) with axis at $45°$ rotates horizontal to right-circular polarization. The output laser pulses are reduced the intensity to a mean photon number (μ) of 0.5 photons per pulse after passing through an iris diaphragm and metallic neutral density (ND). The 8 cm focal length lens forms a collimated beam.

Finally, the light beams of two paths are coupled into a multimode 850 nm fiber, and adjust the fiber optic collimator for a having maximal intensity which measured by the photodiode (shown in figure 5).

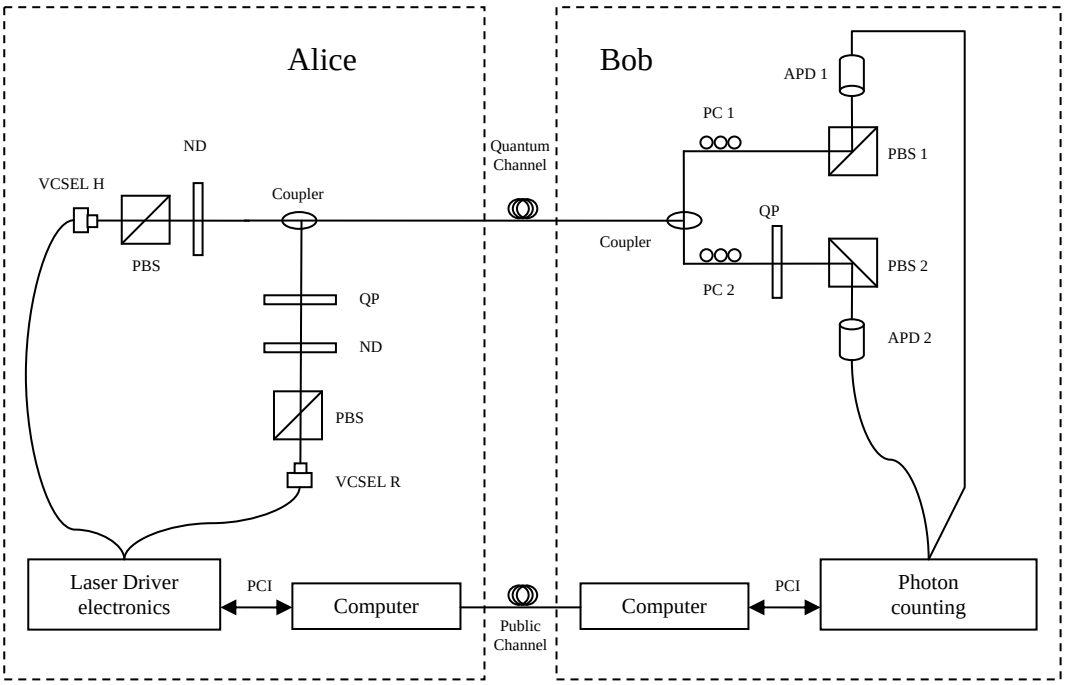

Figure 4 The schematic diagram of our QKD system.

Figure 5 The transmitter module of our QKD system.

The Receiver Section

The receiver setup is constructed to analyze the two photon polarization states from the transmitter.

1. Electronic part

The single photon detectors used are four channel SPCM-AQ4C modules made by PerkinElmer; inside are silicon avalanche photo-diodes operated in Geiger counter mode. These are pn photo-diodes operated with a reverse bias voltage in excess of their breakdown voltage; when a photon strikes the diode, an electron-hole pair is created, which induces a charge avalanche that is electronically transformed into a 25 ns wide TTL pulse output from the detectors. These detectors have an efficiency of ~ 40% at a wavelength of 850 nm, and a dead time of about 50 ns after each detection, as the avalanche must be quenched and the detector reset. The detectors also have a dark count rate of ~ 1000 cps since they are heavily reversed biased so that the thermal excitation of any impurities can cause a charge avalanche and false detection.

The synchronizing clock is sent from the transmitter to coincident electronics (Pornkaveerat, 2008). The coincident electronics search for the rising edge of the photon detection signals from the APDs.

The output from APDs would be simultaneously collected and stored in the memory of the personal computer with the exact arrival time of each photon by the NI PCI-6221 card.

The upper limit of the synchronization input of the NI PCI-6221 card was 250 KHz. Thus, the synchronization frequency, which equal to the laser clock frequency, must be less than 250 KHz per channel. In this experiment, the measurements were only taken at a clock frequency of 10 KHz with a three channels.

2. Optical part

The fiber polarization controller (PC) was installed in each path to recover the photon’s polarization state.

The PC was adjusted so photons from VCSEL H (+0 degrees) have a maximal probability of reaching APD2 and photons from VCSEL R (right-circular) have a maximal probability of reaching APD1.

The cube polarizing beam splitter (PBS1) was installed after the PC1 and install the quarter wave plate (QP), followed the PBS2, after the PC2.

Finally, the light beams of two paths are collimated into a multimode fiber and focused onto the surface of the SPCM-AQ4C for detection (shown in figure 6)

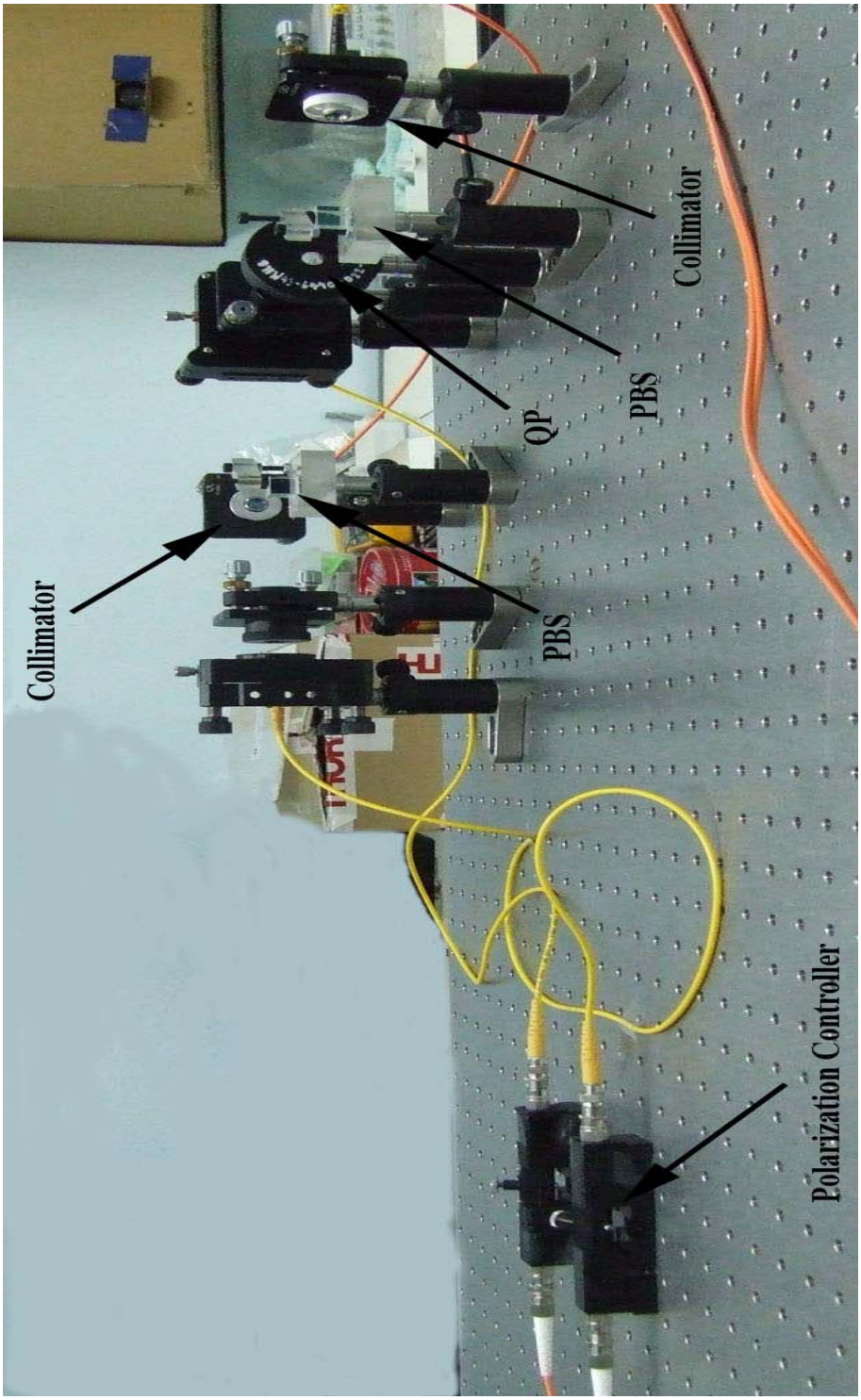

Figure 6 The receiver module of our QKD system.

.

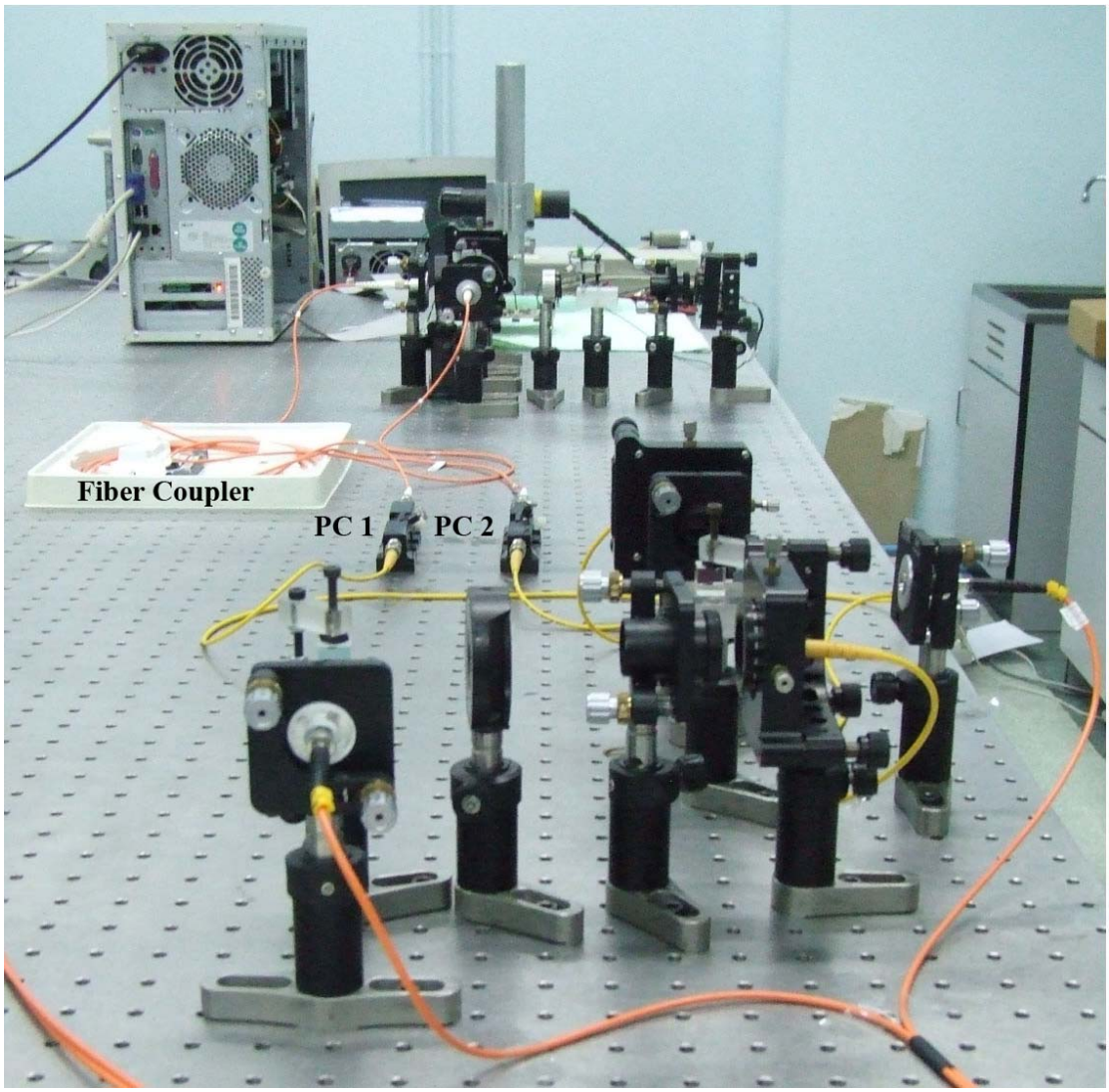

Figure 7 The setup of our fiber based QKD system.

Software

There are a number of different software applications one would like to have available to be used with the QKD system described in this thesis. First, in order to address the VCSELs randomly, a computer program (the flowchart is shown in figure 8 and the screen shot is shown in figure 9) developed by Borland Delphi7 used to control a PCI8255 V3 card. Second, since the series of repetitive data were collected by the NI PCI-6221 card comprised of the transmitter's data and the detector dark counts, so a computer program for analyzing the series of repetitive data (the flowchart is shown in figure 10 and the screen shot is shown in figure 11) was use to discard of the irrelevant bits. Last, an application to perform the B92 protocol was a computer program for comparing the basis (the flowchart is shown in figure 12 and the screen shot is shown in figure 13).

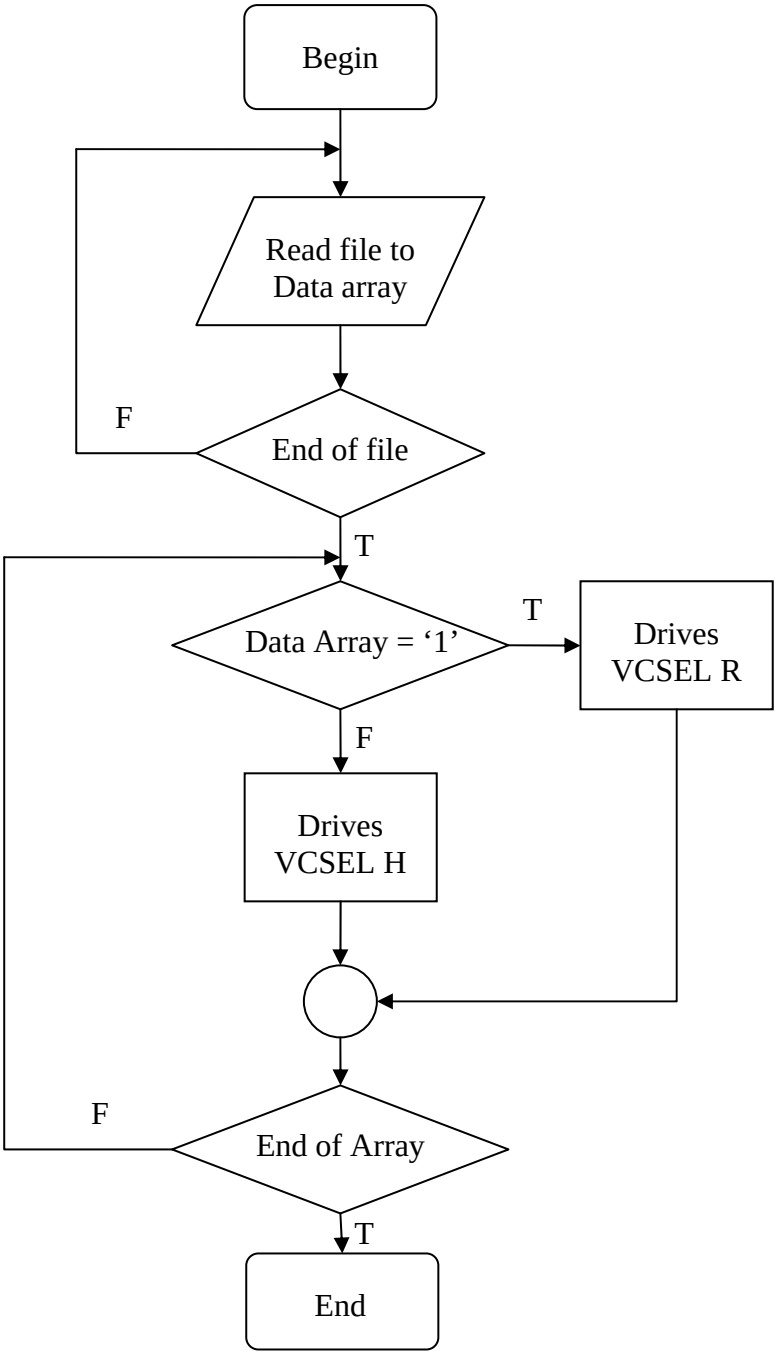

Figure 8 The flowchart of a computer program for controlling PCI card.

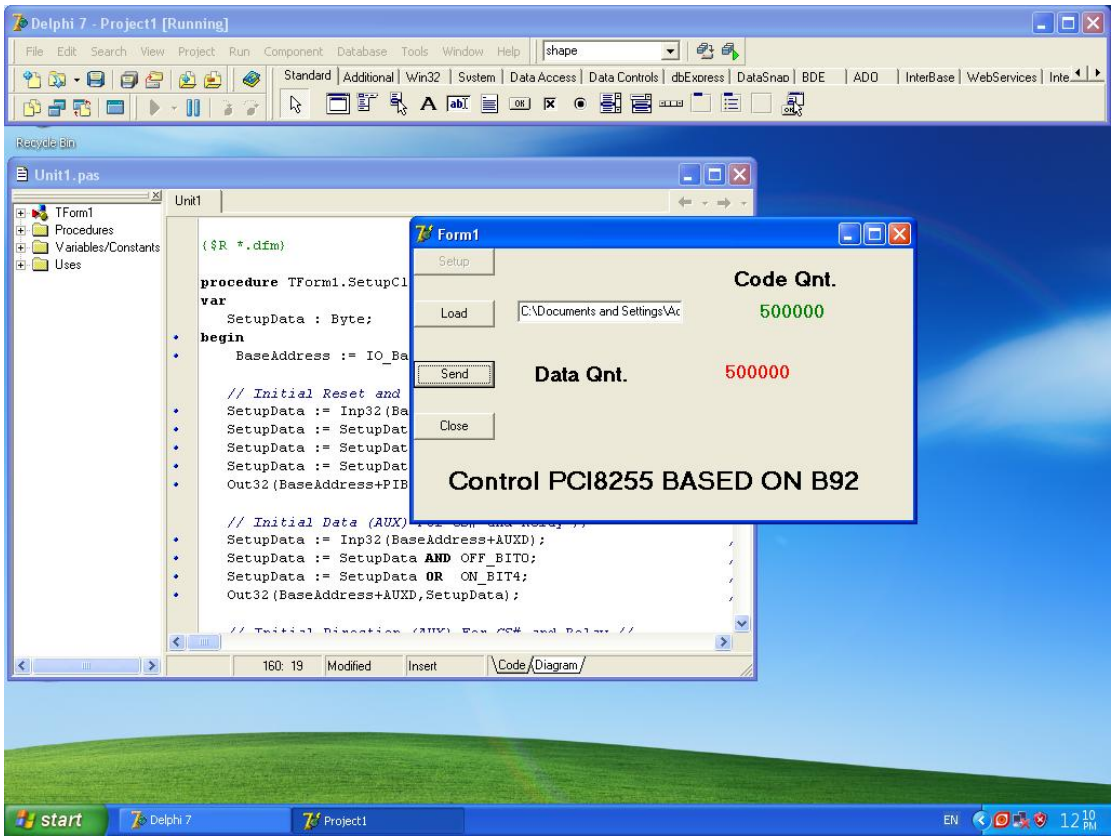

Figure 9 The screen shot of a computer program for controlling PCI card.

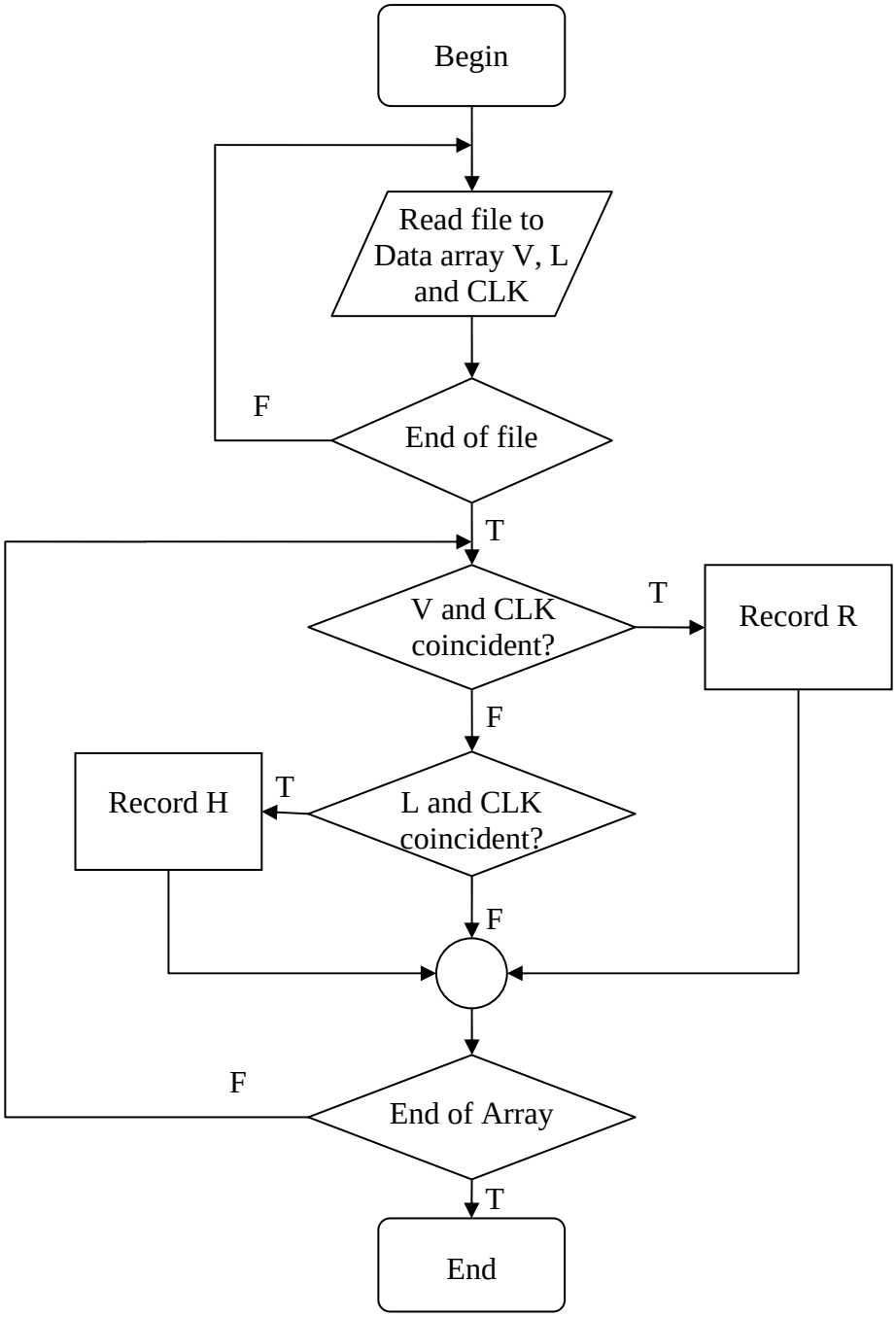

Figure 10 The flowchart of a computer program for analyzing series of raw data.

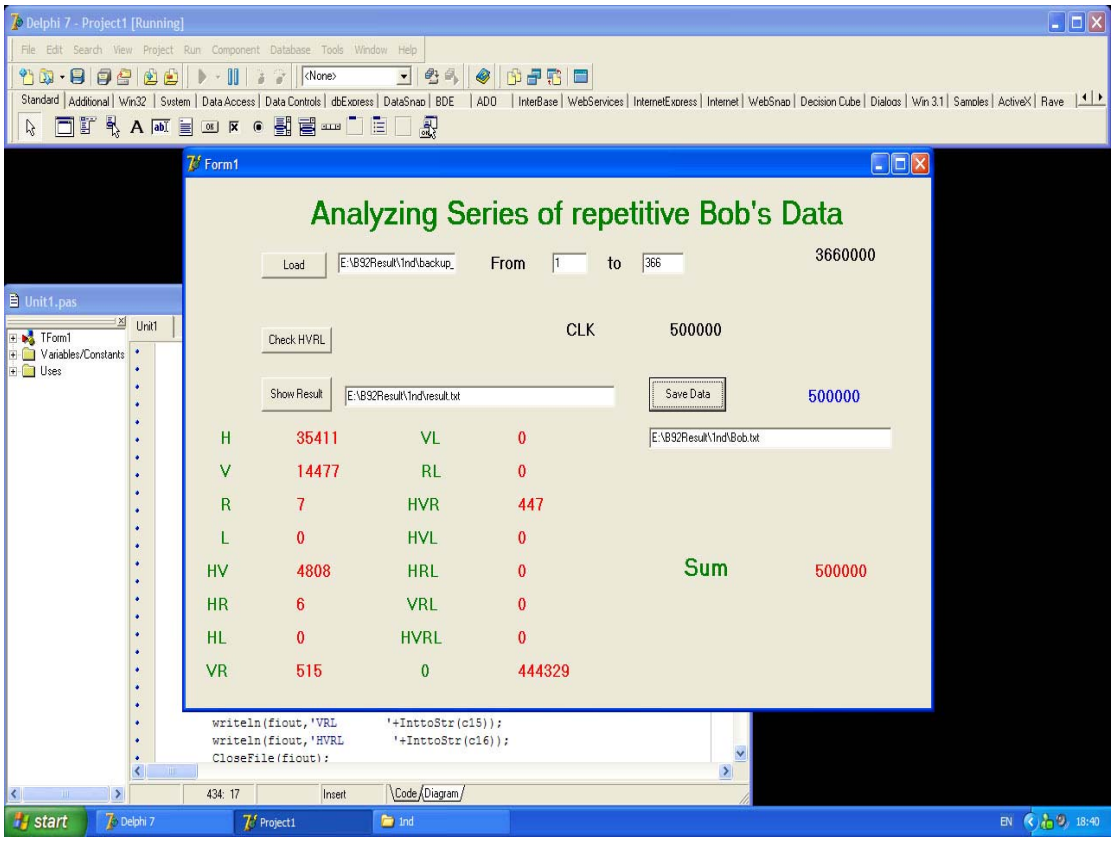

Figure 11 The screen shot of a computer program for analyzing series of raw data.

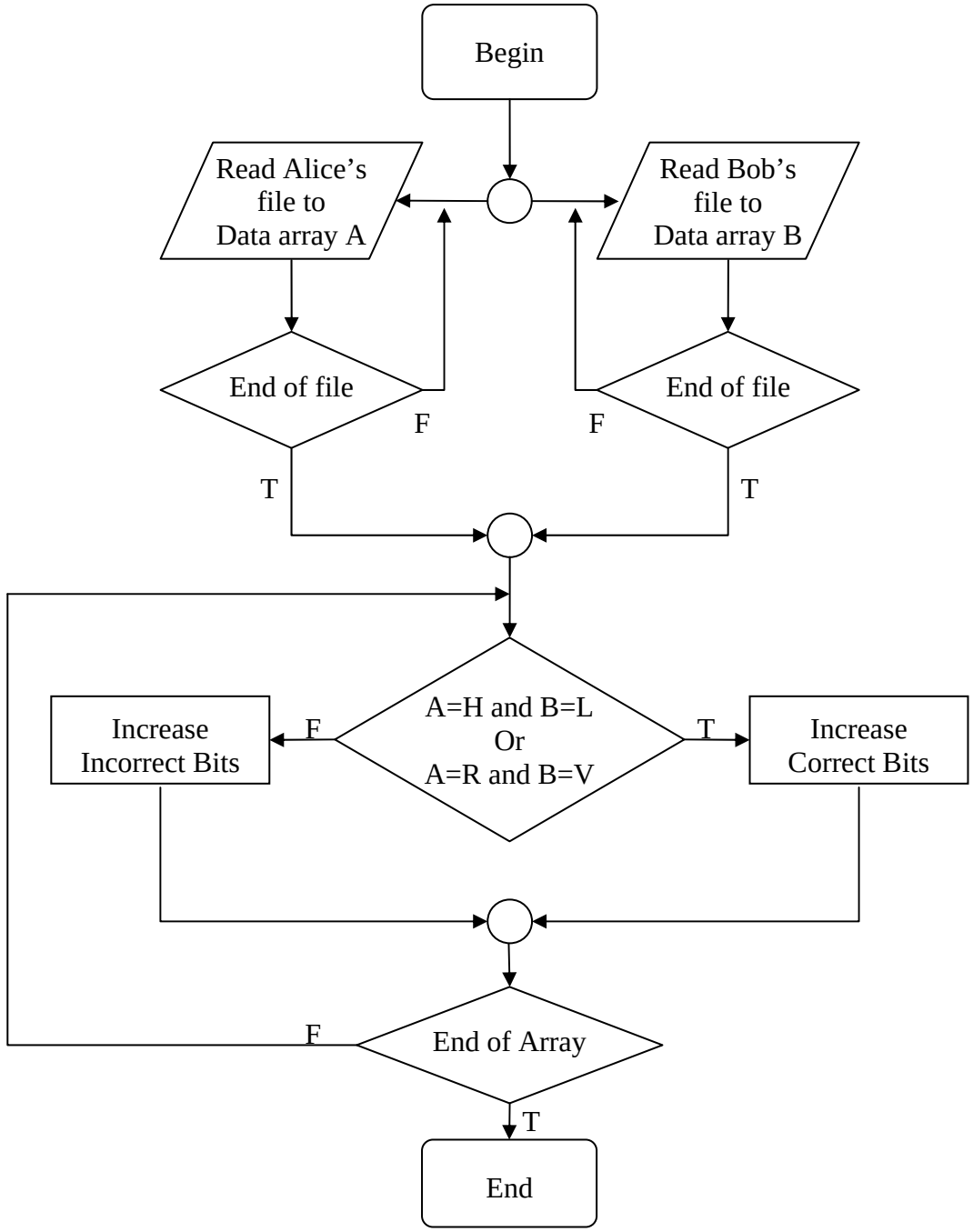

Figure 12 The flowchart of a computer program for comparing between Alice's and Bob's basis.

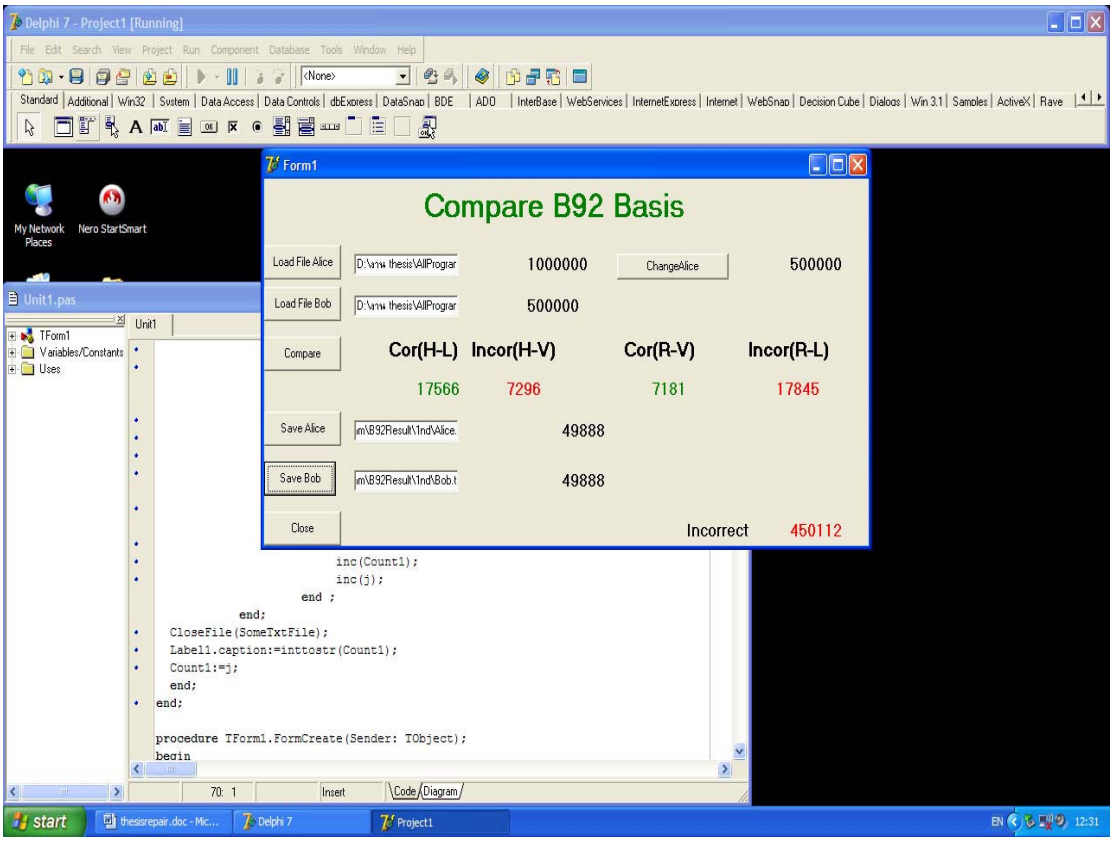

Figure 13 The screen shot of a computer program for comparing between Alice's and Bob's basis.

RESULTS AND DISCUSSION

Results

In this study, we focus on designing, constructing since these quantify system performance for a given sifted-key rate and the QBER, and developing a program computer for the QKD system. Using B92 we transmitted random quantum streams and performed key generation, measuring sifted-key rate and error rates. Our focus here is the limiting effects on the sifted key rate and the QBER.

The Performance Tests of the Photon Polarization States from the Transmitter's Output

The two polarization states are analyzed by rotating a polarizer at transmitter's output and measured intensity by photodiode, present the variation of the observed intensity depending on the angle of the analyzing for the two polarizations. We observe visibility of $H = 0.97$ and $R = 0.99$, which clearly demonstrates the usability of our simple transmitter for low-noise.

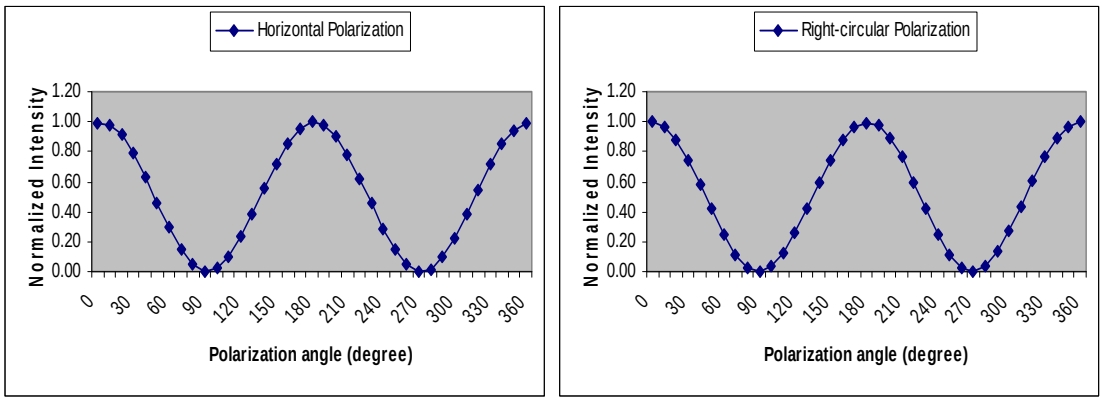

Figure 14 Polarization of a transmitter for the H (left) and R (right).

The Performance Tests of the Photon Polarization States from the Receiver's Output

The polarization states of the photon were recovered by the polarization controllers (PC) which were adjusted so that photons from the transmitter have a maximal probability of reaching APDs. As a result, we can observe visibility of $H = 0.38$ and $R = 0.69$ (shown in figure 15).

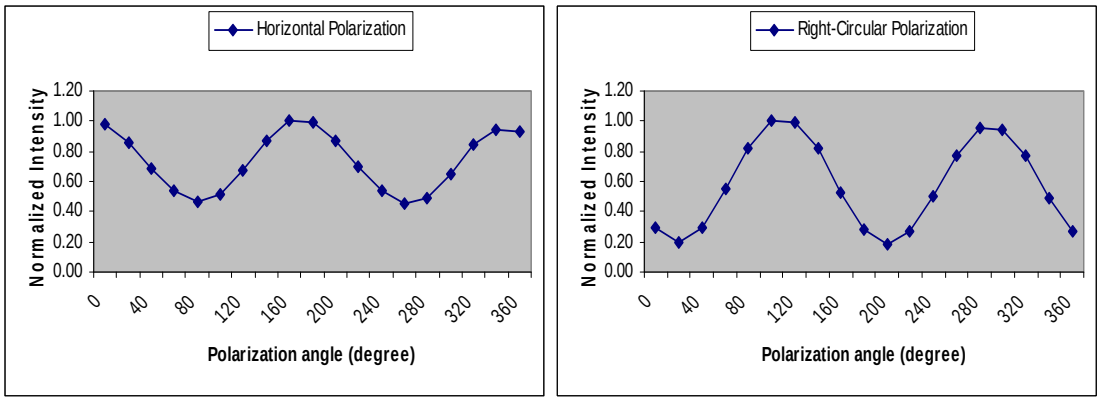

Figure 15 Polarization of a receiver for the H (left) and R (right).

The Sifted-Key Rate and the QBER

Our goal is to evaluate the mean number of polarized photon per pulse, the visibility and synchronization of the system. For each run, the measurements were taken about 50 s of the data acquisition time. A total of 5.0×10^5 photons have been sent, and a total of 4.9×10^4 photons have been recorded after polarized encoding (shown in table 3). The recorded photons after polarized encoding and the data acquisition time show that the sifted-key rate is approximately 1 Kbits/s. We assume that the transmission efficiency of Alice's is 0.9, thus the mean photons number per pulse was $\mu = 0.5$, the frequency of the transmission is 10 KHz, the photon detection efficiency is 0.4, and the factor q can be discard because this experiment did not use a phase-coding setups. We found that the sifted-key rate is 1 Kbits/s calculated with equation (2).

Table 3 The number of photons recorded in NI PCI6221 card.

APD channel	The number of measured signal
APD V	14,477
APD L	35,411
APD V and APD L	4,808
Unable detected	445,304
Total	500,000

Quantum-bit errors are mainly caused by the following: (1) Spontaneous triggering of the APDs (dark counts); (2) Polarization leakage caused by the imperfect polarization extinction ratio; (3) Timing jitter of the system. The experimental data shown that the average QBER of 50.39%. The detector dark count is 1000 counts/s, thus the probability of registering a dark count per time window and per detector (p_{dark}) is 5.0×10^{-5}, and the $QBER_{acc}$ can be discard because this factor appears only in systems based on entangled photons. We found that the average QBER is 50.39% calculated with equation (7), (8) and (9).

Discussion

The visibility of the receiver for the H and R which are 0.38 and 0.69 respectively. This visibility indicates that the average QBER of 50.39%. To achieve a QBER below 1%, the polarization controller must be adjusted so that the photons from the transmitter have a minimal probability of reaching the receiver.

CONCLUSION AND RECOMMENDATION

Conclusion

In this thesis, we have studied the experimental QKD system based on the B92 protocol over the standard telecommunications fiber. Our QKD system consists of a transmitter and a receiver. The transmitter comprises of two VCSELs, controllable to switch on-off one for all by a PCI8255 V3 card. The experimental data shown that a mean photon number (μ) of 0.5 photons per pulse. The optical setup of the transmitter comprises of two polarizing beam-splitter and a quarter wave plate, to assign the polarization states of photons according to B92 protocol, there are two polarization states of photons.

Our receiver contains of SPCM-AQ4C to detect the single photons and an optical setup, which consist of two polarizing beam-splitter, two polarization controller and a quarter wave plate, to recover and analyze the polarization states of photons from the transmitter. We observed the visibilities of the horizontal and right-circular polarization states of the single photons coming from our transmitter are measured with photodiode (DET210) are 0.99 and 0.97 respectively. The visibility of the receiver for the H and R polarization states of the single photons which are 0.38 and 0.69 respectively.

The QKD system was operated over 1-m fiber range indoor optical part, and they are located on an optical breadboard. We evaluate the sifted-key rate is 1 Kbits/s and the average QBER of 50.39%.

Recommendation

In the further studies, we will adjust the polarization controller so that the photons from the transmitter have a minimal probability of reaching the receiver, and try to increase the visibility of the receiver. Further more, we will apply the optical fiber for QKD in another protocol and increase span length .

LITERATURE CITED

Bennett, C.H. 1992. Quantum cryptography using any two non-orthogonal states. **Phys. Rev. Lett.** 68 (21): 3121-3124.

_____ and G. Brassard. 1984. Quantum cryptography: Public key distribution and coin tossing, pp. 175-179. *In* **Proceedings of IEEE International Conference on Computers, Systems and Signal Processing**. Bangalore, India.

_____, _____, and N.D. Mermin. 1992. Quantum cryptography without Bell's theorem. **Phys. Rev. Lett.** 68: 557.

_____, F. Bessette, G. Brassard, L. Salvail, and J. Smolin. 1992. Experimental quantum cryptography. **J. of Cryptology**. 5: 3.

Benthune, D.S., M. Navarro and W.P. Risk. 2002. Enhanced autocompensating quantum cryptography system. **Appl. Opt**. 41: 1640-1648.

Beveratos, A., R. Brouri, T. Gacoin, A. Villing, J.P. Poizat, and P. Grangier. 2002. Single photon quantum cryptography. **Phys. Rev. Lett.** 89: 187901

Biengfang, J.C., A.J. Gross, A. Mink, B.J. Hershman, A. Nakassis, X. Tang, R. Lum, D.H. Su and C.W. Clark. 2004. Quantum key distribution with 1.25 Gbps clock synchronization. **Opt. Express**. 7: 2011-2016.

Breguet, J., A. Muller and N. Gisin. 1994. Quantum cryptography with polarized photons in optical fibers, experiment and practical limits. **J. of Mod. Opt.** 41: 2405-2412.

Deachapunya, S. 2002. **Experimental Quantum Cryptography Based on the BB84 Protocol.** M.S. Thesis, Kasetsart University.

Ekert, A.K. 1991. Quantum cryptography based on Bell's theorem. **Phys. Rev. Lett.** 67: 661.

Elliott, C., D. Pearson and G. Troxel. 2003. Quantum cryptography in practice, pp. 227-238. *In* **SIGCOMM' 03: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications**. ACM Press, New York.

Erven, C. 2007. **On Free Space Quantum Key Distribution and its Implementation with a Polarization-Entangled Parametric Down Conversion Source**. M.S. Thesis, University of Waterloo.

Gisin, N., G. Ribordy, W. Tittel and H. Zbinden. 2002. Quantum cryptography. **Rev. Mod. Phys.** 74: 145-195.

Gordon, K.J., V. Fernandez, P.D. Townsend and G.S. Buller. 2004. A short wavelength GigaHertz clocked fiber optic quantum key distribution system. **IEEE J. of Quantum Electron**. 40: 900-908.

Hughes, R.J., G.L. Morgan and C.G. Peterson. 2000. Quantum key distribution system over a 48 Km optical fiber network. **J. Mod. Opt**. 47: 533-547.

Hwang, W.Y. 2003. Quantum key distribution with high loss: Toward global secure communication. **Phys. Rev. Lett.** 91: 057901

Lindenthal, M. 2006. **Long-Distance Free-Space Quantum Communication with Entangled Photons**. Ph.D. Thesis, University of Vienna.

Lodewyck, J., R.G. Patron, S. Fossier, E. Karpov, E. Diamanti, T. Debuisschert, N.J. Cerf, R.T. Brouri, S.W. McLaughlin and P. Grangier. 2007. Quantum key distribution over 25 Km with an all-fiber continuous-variable system. **Phys. Rev. A**. 76 (042305): 1-10.

Lomonaco, S.J.Jr. 2009. **Quantum computation & quantum information**. QCryptoHandout. Available Source: http://www.csee.umbc.edu/~lomonaco/, February 3, 2009.

Panthong, P. 2005. **The Performance of Quantum Cryptography System**. M.S. Thesis, Kasetsart University.

Pirandola, S., S.L. Braunstein and S. Lloyd. 2008. Characterization of collective Gaussian attacks and security of coherent-state quantum cryptography. Phys**. Rev. Lett**. 101: 200504.

Pornkaveerat, W. 2009. **Experimental Quantum Cryptography Based on the SARG04 Protocol**. M.S. Thesis, Kasetsart University.

Rarity, J.G., P.R. Tapster and P.M. Gorman. 2001. Secure free-space key-exchange to 1.9 Km and beyond. **J. Mod. Opt.** 48: 1887-1901.

Shannon, C.E. 1949. Communication theory of secrecy systems. **Bell Syst. Tech. J.** 28: 656-715.

Tang, X., L. Ma, A. Mink, A. Nakassis, H. Xu, B. Hershman, J.C. Bienfang, D. Su, R.F. Boisvert, C.W. Clark and C.J. Williams. 2006. Experimental study of high speed polarization-coding quantum key distribution with sift-key rates over Mbit/s. **Opt. Express.** 14: 2062-2070.

Townsend, P.D. 1998. Experimental investigation of the performance limits for first telecommunication-window quantum cryptography system. **IEEE Photon. Technol. Lett.** 10: 1048-1050.

APPENDIX

The Computer Program Developed by Borland Delphi7

The Computer Program for Control the PCI8255 V3 Card

1. I/O address of ET-PCI8255 V3

```
Const  IO_BaseAddress = $E800;     // I/O Base Address
```

2. Tiger-320 register offset

```
Const  PIB  = $00;                 // Reset & PIB Cycle
Const  AUXC = $02;                 // Aux Direction Port
Const  AUXD = $03;                 // Aux Data Port
Const  PA1  = $C0;                 // Port-A 8255#1
Const  PB1  = $C4;                 // Port-B 8255#1
Const  PC1  = $C8;                 // Pprt-C 8255#1
Const  PCC1 = $CC;                 // Port Control 8255#1
Const  PA2  = $D0;                 // Port-A 8255#2
Const  PB2  = $D4;                 // Port-B 8255#2
Const  PC2  = $D8;                 // Port-C 8255#2
Const  PCC2 = $DC;                 // Port Control 8255#2
Const  PA3  = $E0;                 // Port-A 8255#3
Const  PB3  = $E4;                 // Port-B 8255#3
Const  PC3  = $E8;                 // Port-C 8255#3
Const  PCC3 = $EC;                 // Port Control 8255#3
Const  ON_Bit0   = $01;            // XXXX XXXX OR  0000 0001 =
XXXX XXX1
Const  OFF_Bit0  = $FE;            // XXXX XXXX AND 1111 1110 =
XXXX XXX0
Const  ON_Bit1   = $02;            // XXXX XXXX OR  0000 0010 =
XXXX XX1X
```

```
        Const  OFF_Bit1  = $FD;          // XXXX XXXX AND 1111 1101 =
XXXX XX0X
        Const  ON_Bit2   = $04;          // XXXX XXXX OR  0000 0100 =
XXXX X1XX
        Const  OFF_Bit2  = $FB;          // XXXX XXXX AND 1111 1011 =
XXXX X0XX
        Const  ON_Bit3   = $08;          // XXXX XXXX OR  0000 1000 =
XXXX 1XXX
        Const  OFF_Bit3  = $F7;          // XXXX XXXX AND 1111 0111 =
XXXX 0XXX
        Const  ON_Bit4   = $10;          // XXXX XXXX OR  0001 0000 =
XXX1 XXXX
        Const  OFF_Bit4  = $EF;          // XXXX XXXX AND 1110 1111 =
XXX0 XXXX
        Const  ON_Bit5   = $20;          // XXXX XXXX OR  0010 0000 =
XX1X XXXX
        Const  OFF_Bit5  = $DF;          // XXXX XXXX AND 1101 1111 =
XX0X XXXX
        Const  ON_Bit6   = $40;          // XXXX XXXX OR  0100 0000 =
X1XX XXXX
        Const  OFF_Bit6  = $BF;          // XXXX XXXX AND 1011 1111 =
X0XX XXXX
        Const  ON_Bit7   = $80;          // XXXX XXXX OR  1000 0000 =
1XXX XXXX
        Const  OFF_Bit7  = $7F;          // XXXX XXXX AND 0111 1111 =
0XXX XXXX
        Const  OutputOFF = ClMenu;       // Color of Output OFF ("0")  Status
        Const  OutputON  = ClRed;        // Color of Output ON  ("1")  Status
        Const  InputOFF  = ClBlue;       // Color of Input  OFF ("1")  Status
        Const  InputON   = ClMenu;       // Color of Input  ON  ("0")  Status
```

3. Initial values for PCI8255 V3 part.

```
Var
        SetupData : Byte;

Begin
        BaseAddress := IO_BaseAddress;              // Set I/O Base Address

// Initial Reset and 8255 Bus Cycle //
        SetupData := Inp32(BaseAddress+PIB);        // Read PIB Reset Port
        SetupData := SetupData AND OFF_BIT0;        // Bit0 = EXTRST# = "0"
(Reset:RES#)
        SetupData := SetupData OR ON_BIT5;          // Bit5:4 = 11 = PIB Cycle
Slowest
        SetupData := SetupData OR ON_BIT4;
        Out32(BaseAddress+PIB,SetupData);           // Active RES# & Relay

// Initial Data (AUX) For CS# and Relay //
        SetupData := Inp32(BaseAddress+AUXD);       // Read Aux Data Port
        SetupData := SetupData AND OFF_BIT0;        // Bit0 = Aux0 = "0" (Enable
CS)
        SetupData := SetupData OR  ON_BIT4;         // Bit4 = Aux4 = "1" (Relay
OFF)
        Out32(BaseAddress+AUXD,SetupData);          // Active Chips Select &
Relay

// Initial Direction (AUX) For CS# and Relay //
        SetupData := Inp32(BaseAddress+AUXC);       // Read Aux Port Direction
        SetupData := SetupData OR ON_BIT4;          // Aux4 = "1" = Output
        SetupData := SetupData OR ON_BIT0;          // Aux0 = "1" = Output
        Out32(BaseAddress+AUXC,SetupData);          // Setup Aux Direction
```

```
// Initial 8255#1 = All Input Port //
        Setup.Enabled := False;                    // Disable Setup After Setup
        Out32(BaseAddress+PCC1,$80);               // Write Control Port 8255#1
End;
```

4. Reading file to data array part.

```
Var
        SomeTxtFile:textfile;
        col,j:integer;

Begin
        if OpenDialog1.Execute then
        Begin
            Edit1.text:=opendialog1.Filename;
            AssignFile(SomeTxtFile,OpenDialog1.FileName);
            reset(SomeTxtFile);
            j:=1;row:=0;
            while not EOF(SomeTxtFile) do
            Begin
                    col:=1;
                    While (col<25) And not EOF(SomeTxtFile) do
                    Begin
                        inc(col);
                        inc(j);
                        inc(Count);
                        Read(SomeTxtFile, OneData[j]);
                    End;
                    If (col=25) And not EOF(SomeTxtFile) then
                    Begin
                        inc(j);
                        inc(Count);
```

```
                    Readln(SomeTxtFile, OneData[j]);
                End;
            End;
        End;
        CloseFile(SomeTxtFile);
        Label1.Caption:=InttoStr(Count);
End;
```

5. Addressing the VCSELs part.

```
Var
        i,sum:integer;

Begin
        sum:=0;
        For i:=1 to Count do
        Begin
            if (OneData[i]='0') then
            Begin
                Out32(BaseAddress+PA1,$03); // 0000 0011=0:L:0:R 0:V:CLK:H
                Out32(BaseAddress+PA1,$00);
            End
            Else if (OneData [i]='1') then
            Begin
                Out32(BaseAddress+PA1,$12); // 0001 0010=0:L:0:R 0:V:CLK:H
                Out32(BaseAddress+PA1,$00);
            End
            inc(sum);
        End;
        Label2.Caption:=InttoStr(sum);
End;
```

The Computer Program for Analyzing Series of Repetitive Data (B92 Protocol)

1. Initial variable part

```
Var
    Count,Bit,j:integer;
    DataV: Array[1..8000000] of Real;
    DataL: Array[1..8000000] of Real;
    DataCLK: Array[1..8000000] of Real;
    Data2: Array[1..8000000] of String;
```

2. Read 3 column received data to data array part

```
Var
    t:integer;
    SomeTxtFile:textfile;
    p,q,r,s,FileName:String;

Begin
    p:=Edit1.Text;
    q:=Edit4.Text;
    r:=Edit5.Text;
    For t:=StrtoInt(q) to StrtoInt(r) do
    Begin
        s:=InttoStr(t);
        FileName:=p+s+'.txt';
        AssignFile(SomeTxtFile, FileName);
        reset(SomeTxtFile);
    While not EOF(SomeTxtFile) do
    Begin
        inc(Count);
        Read(SomeTxtFile, DataV[Count]);
```

```
            Read(SomeTxtFile, DataV[Count]);
            Read(SomeTxtFile, DataL[Count]);
            Read(SomeTxtFile, DataL[Count]);
            Read(SomeTxtFile, DataCLK[Count]);
            Readln(SomeTxtFile, DataCLK[Count]);
            inc(Bit);
        End;
        CloseFile(SomeTxtFile);
        Label1.Caption:=InttoStr(Bit);
        End;
End;
```

3. Check the synchronized data with clock part

```
Var
        i,check,k1,k2:integer;

Begin
        check:=0; k1:=0; k2:=0;
        For i:=1 to (bit+1) do
        Begin
            if (DataCLK[i]>2) then
            Begin
                    inc(check);
                    if check=2 then k1:=i;
                    if check=3 then k2:=i;
            End;
            if (DataCLK[i]<2) then
            Begin
                    If not odd(Check) and (check>0) then
                    Begin
```

```
                if (DataV[k1]<2) and (DataL[k1]<2) and (DataCLK[k1]>2)
then
                Begin
                        inc(j);
                        Data2[j]:='N';    //0
                        inc(c1);
                End;
                if (DataV[k1]>2) and (DataL[k1]<2) and (DataCLK[k1]>2)
then
                Begin
                        inc(j);
                        Data2[j]:='V';    //V
                        inc(c2);
                End;
                if (DataV[k1]<2) and (DataL[k1]>2) and (DataCLK[k1]>2)
then
                Begin
                        inc(j);
                        Data2[j]:='L';    //L
                        inc(c3);
                End;
                if (DataV[k1]>2) and (DataL[k1]>2) and (DataCLK[k1]>2)
then
                Begin
                        inc(j);
                        Data2[j]:='N';    //VL
                        inc(c4);
                End;
        End;
        If  odd(check) and (check>0) then
        Begin
```

```
                    if (DataV[k1]<2) and (DataL[k1]<2) and (DataCLK[k1]>2)
        then
                    Begin
                        inc(j);
                        Data2[j]:='N';    //0
                        inc(c1);
                    End;
                    if (DataV[k1]>2) and (DataL[k1]<2) and (DataCLK[k1]>2)
        then
                    Begin
                        inc(j);
                        Data2[j]:='V';    //V
                        inc(c2);
                    End;
                    if (DataV[k1]<2) and (DataL[k1]>2) and (DataCLK[k1]>2)
        then
                    Begin
                        inc(j);
                        Data2[j]:='L';    //L
                        inc(c3);
                    End;
                    if (DataV[k1]>2) and (DataL[k1]>2) and (DataCLK[k1]>2)
        then
                    Begin
                        inc(j);
                        Data2[j]:='N';    //VL
                        inc(c4);
                    End;
                End;
          End;
End;
```

4. Collect data to Bob's file part

```
Var
        n,l,sumn:integer;
        fiout:textfile;
        filename,RowData:string;

Begin
        sumn:=c1+c2+c3+c4;
        n:=0;
        filename:=Edit2.text;
        rewrite(fiout,filename);
        while n< j do
        Begin
            RowData:='';
            l:=0;
            while (n<j) And (l<25) do
                    Begin
                        inc(n);
                        inc(l);
                        RowData:=RowData+Data2[n];
                    End;
            writeln(fiout,RowData);
        End;
        CloseFile(fiout);
End;
```

The Computer Program for Comparing the Basis (B92 Protocol)

```
Var
        p,n,Incor:integer;
```

```
Begin
        p:=1;n:=1;Incor:=0;
        While p < (m+1) do
        Begin
            If ((Data1c[p]='H') and (Data2[p]='H')) then
            Begin
                    Data3[n]:=Data1c[p];
                    Data4[n]:=Data2[p];
                    inc(p);
                    inc(n);
                    inc(Cor1);
            End
            Else if ((Data1c[p]='R') and (Data2[p]='V')) then
            Begin
                    Data3[n]:=Data1c[p];
                    Data4[n]:=Data2[p];
                    inc(p);
                    inc(n);
                    inc(Cor2);
            End
            Else if ((Data1c[p]='H') and (Data2[p]='V')) then
            Begin
                    Data3[n]:=Data1c[p];
                    Data4[n]:=Data2[p];
                    inc(p);
                    inc(n);
                    inc(Cor3);
            End
            Else if ((Data1c[p]='R') and (Data2[p]='H')) then
            Begin
                    Data3[n]:=Data1c[p];
                    Data4[n]:=Data2[p];
```

```
            inc(p);
            inc(n);
            inc(Cor4);
        End
        Else
        Begin
            inc(p);
            inc(Incor);
        End;
End;
```

The Program Computer for Comparing Bit Value between Bob and Alice

1. Initial variable part

```
Var
    OneData1,OneData2:Array[1..8000000] of Char;
    Count,Count2:integer;
```

2. Read Alice's file to data array part

```
Var
    j,col:integer;
    SomeTxtFile:textfile;

Begin
    if OpenDialog1.Execute then
    Begin
        Edit1.text:=opendialog1.Filename;
        AssignFile(SomeTxtFile,OpenDialog1.FileName);
        reset(SomeTxtFile);
        j:=0;Count:=0 ;
```

```
        While not EOF(SomeTxtFile) do
    Begin
        col:=1;
        While (col<25) And not EOF(SomeTxtFile) do
        Begin
            inc(col);
            inc(j);
            inc(Count);
            Read(SomeTxtFile, OneData1[j]);
        End;
        if (col=25) And not EOF(SomeTxtFile) then
        Begin
            inc(j);
            inc(Count);
            Readln(SomeTxtFile, OneData1[j]);
        End;
    End;
    End;
End;
```

3. Read Bob's file to data array part

```
Var
    j,col:integer;
    SomeTxtFile:textfile;

Begin
    if OpenDialog2.Execute then
    Begin
        Edit1.text:=opendialog2.Filename;
        AssignFile(SomeTxtFile,OpenDialog2.FileName);
        reset(SomeTxtFile);
```

```
        j:=0;Count2:=0 ;
        While not EOF(SomeTxtFile) do
    Begin
        col:=1;
        While (col<25) And not EOF(SomeTxtFile) do
        Begin
            inc(col);
            inc(j);
            inc(Count2);
            Read(SomeTxtFile, OneData2[j]);
        End;
        if (col=25) And not EOF(SomeTxtFile) then
        Begin
            inc(j);
            inc(Count2);
            Readln(SomeTxtFile, OneData2[j]);
        End;
    End;
    End;
End;
```

4. Compare bit value part

```
Var
    i,CountError,CountCorrect:integer;

Begin
    CountError:=0;
    CountCorrect:=0;
    If Count<>Count2 Then Label8.Caption:='Error'
    Else
    Begin
```

```
        For i:=1 to Count do
        Begin
            If OneData1[i]<>OneData2[i] Then inc(CountError)
            else  inc(CountCorrect);
        End;
    End;
End;
```

The Computer Program for Addressing the VCSELs (SARG04 Protocol)

1. Initial variable part

```
Var
    BaseAddress : Word;
    Data1,Data2,Data3: array[1..8000000] of char;
    Count:integer;
    Procedure Out32(Port:Word; Data:Byte); Stdcall;External'InpOut32.DLL';
    Function  Inp32(Port:Word):Byte; Stdcall; External 'InpOut32.DLL';
```

2. Reading file to data array part

```
Var

    SomeTxtFile:textfile;
    col,row,j,n:integer;

Begin
    if OpenDialog1.Execute then
    Begin
        Edit1.text:=opendialog1.Filename;
        AssignFile(SomeTxtFile,OpenDialog1.FileName);
        reset(SomeTxtFile);
```

```
j:=1;row:=0;n:=0;
While not EOF(SomeTxtFile) do
Begin
        col:=1;
        inc(row);
        If row = (1+(3*n)) then
        Begin
            While (col<25) And not EOF(SomeTxtFile) do
            Begin
                    Read(SomeTxtFile, Data1[j]);
                    inc(col);
                    inc(Count);
            If not EOF(SomeTxtFile) then
            Begin
                    Read(SomeTxtFile, Data2[j]);
                    inc(Count);
                    inc(col);
            End;
            If not EOF(SomeTxtFile) then
            Begin
                    Read(SomeTxtFile, Data3[j]);
                    inc(Count);
                    inc(col);
                    inc(j);
            End;
        End;
        If (col=25) and not EOF(SomeTxtFile) then
        Begin
            Readln(SomeTxtFile, Data1[j]);
            inc(Count);
        End;
End;
```

```
If row = (2+(3*n)) then
Begin
        While (col<3) And not EOF(SomeTxtFile) do
        Begin
                Read(SomeTxtFile, Data2[j]);
                inc(col);
                inc(Count);
                If not EOF(SomeTxtFile) then
                Begin
                        Read(SomeTxtFile, Data3[j]);
                        inc(Count);
                        inc(col);
                        inc(j);
                End;
        End;
        While (col>2) And (col<24) And not EOF(SomeTxtFile) do
        Begin
                Read(SomeTxtFile, Data1[j]);
                inc(col);
                inc(Count);
                if not EOF(SomeTxtFile) then
                Begin
                        Read(SomeTxtFile, Data2[j]);
                        inc(Count);
                        inc(col);
                End;
        If not EOF(SomeTxtFile) then
        Begin
                Read(SomeTxtFile, Data3[j]);
                inc(Count);
                inc(col);
                inc(j);
```

```
        End;
        End;
        If (col>23) And (col<26) And not EOF(SomeTxtFile) then
        Begin
            Read(SomeTxtFile, Data1[j]);
            inc(col);
            inc(Count);
            If not EOF(SomeTxtFile) then
            Begin
                Readln(SomeTxtFile, Data2[j]);
                inc(Count);
                inc(col);
            End;
        End;
End;
If row = (3+(3*n)) then
Begin
        if (col=1) And not EOF(SomeTxtFile) then
        Begin
            Read(SomeTxtFile, Data3[j]);
            inc(col);
            inc(Count);
            inc(j);
        End;
        While (col>1) And (col<23) And not EOF(SomeTxtFile) do
        Begin
            Read(SomeTxtFile, Data1[j]);
            inc(col);
            inc(Count);
            If not EOF(SomeTxtFile) then
            Begin
                Read(SomeTxtFile, Data2[j]);
```

```
                    inc(Count);
                    inc(col);
                End;
                If not EOF(SomeTxtFile) then
                Begin
                    Read(SomeTxtFile, Data3[j]);
                    inc(Count);
                    inc(col);
                    inc(j);
                End;
            End;
            If (col>22) And (col<26) And not EOF(SomeTxtFile) then
            Begin
                Read(SomeTxtFile, Data1[j]);
                inc(col);
                inc(Count);
                If not EOF(SomeTxtFile) then
                Begin
                    Read(SomeTxtFile, Data2[j]);
                    inc(Count);
                    inc(col);
                End;
                If not EOF(SomeTxtFile) then
                Begin
                    Readln(SomeTxtFile, Data3[j]);
                    inc(Count);
                    inc(col);
                    inc(j);
                End;
        End;
inc(n);
End;
```

```
        End;
        CloseFile(SomeTxtFile);
        End;
End;
```

3. Addressing the VCSELs part

```
Var
        i,sum:integer;

Begin
        sum:=0;
        For i:=1 to Count do
        Begin
            If (Data1[i]='0') and (Data2[i]='0') and (Data3[i]='0') then
            Begin
                Out32(BaseAddress+PA1,$03); // 0000 0011=0:L:0:R 0:V:CLK:H
                Out32(BaseAddress+PA1,$00);
            End
            Else if (Data1[i]='0') and (Data2[i]='0') and (Data3[i]='1') then
            Begin
                Out32(BaseAddress+PA1,$03); // 0000 0011=0:L:0:R 0:V:CLK:H
                Out32(BaseAddress+PA1,$00);
            End
            Else if (Data1[i]='0') and (Data2[i]='1') and (Data3[i]='0') then
            Begin
                Out32(BaseAddress+PA1,$06); // 0000 0110=0:L:0:R 0:V:CLK:H
                Out32(BaseAddress+PA1,$00);
            End
            Else if (Data1[i]='0') and (Data2[i]='1') and (Data3[i]='1') then
            Begin
                Out32(BaseAddress+PA1,$06); // 0000 0110=0:L:0:R 0:V:CLK:H
```

```
                Out32(BaseAddress+PA1,$00);
            End
            Else if (Data1[i]='1') and (Data2[i]='0') and (Data3[i]='0') then
            Begin
                Out32(BaseAddress+PA1,$12); // 0001 0010=0:L:0:R 0:V:CLK:H
                Out32(BaseAddress+PA1,$00);
            End
            Else if (Data1[i]='1') and (Data2[i]='1') and (Data3[i]='0') then
            Begin
                Out32(BaseAddress+PA1,$12); // 0001 0010=0:L:0:R 0:V:CLK:H
                Out32(BaseAddress+PA1,$00);
            End
            Else if (Data1[i]='1') and (Data2[i]='0') and (Data3[i]='1') then
            Begin
                Out32(BaseAddress+PA1,$42); // 0100 0010=0:L:0:R 0:V:CLK:H
                Out32(BaseAddress+PA1,$00);
            End
            Else if (Data1[i]='1') and (Data2[i]='1') and (Data3[i]='1') then
            Begin
                Out32(BaseAddress+PA1,$42); // 0100 0010=0:L:0:R 0:V:CLK:H
                Out32(BaseAddress+PA1,$00);
            End;
            inc(sum);
        End;
End;
```

The Program Computer for Comparing the Basis (SARG04)

```
Var
    p,i,Incor:integer;

Begin
```

```
p:=1;n:=1;Incor:=0;
While p < m do
Begin
    If (Data1c[p]='1') and (Data1c[p+1]=Data1[p+1]) then
    Begin
        For i:=1 to 3 do
        Begin
            If i=1 then
            Begin
                Data3[n]:=Data1[p];
                Data4[n]:=Data1c[p];
            End;
            If i=2 then
            Begin
                Data3[n+1]:=Data1[p+1];
                Data4[n+1]:=Data1c[p+1];
            End;
            If i=3 then
            Begin
                Data3[n+2]:=Data1[p+2];
                Data4[n+2]:=Data1[p+2];
            End;

        End;
        p:=p+3;
        n:=n+3;
        inc(Cor);
    End ;
    If (Data1c[p]='0') and (Data1c[p+2]=Data1[p+2]) then
    Begin
        For i:=1 to 3 do
        Begin
```

```
                    If i=1 then
                    Begin
                        Data3[n]:=Data1[p];
                        Data4[n]:=Data1c[p];
                    End ;
                    If i=2 then
                    Begin
                        Data3[n+1]:=Data1[p+1];
                        Data4[n+1]:=Data1[p+1];
                    End;
                    If i=3 then
                    Begin
                        Data3[n+2]:=Data1[p+2];
                        Data4[n+2]:=Data1c[p+2];
                    End ;
                End;
                p:=p+3;
                n:=n+3;
                inc(Cor);
            End
            Else
            Begin
                p:=p+3;
                inc(Incor);
            End;
        End;
End;
```

CURRICULUM VITAE

NAME : Mr. Santhad Phithakwongsaphorn

BIRTH DATE : September 21, 1981

BIRTH PLACE : Ratchaburi, Thailand

EDUCATION :	**YEAR**	**INSTITUTION**	**DEGREE/DIPLOMA**
	2005	Kasetsart Univ.	B.Sc. (Physics)

SCHOLARSHIP/AWARDS : DPST Scholarship 2001-2008