NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

QUANTUM KEY DISTRIBUTION LABORATORY DEMONSTRATION

by

Jack R. Brault

December 2021

Thesis Advisor: Frank A. Narducci
Second Reader: Jihane Mimih

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
QUANTUM KEY DISTRIBUTION LABORATORY DEMONSTRATION

5. FUNDING NUMBERS

6. AUTHOR(S) Jack R. Brault

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Quantum key distribution (QKD) is a method of secure key distribution which provides protection against the tampering and interception of information.

Following the Bennet-Brassard 1984 (BB84) protocol of QKD, we select randomly from a set of bases in which to produce polarized photons and send the photons to a receiver, who measures them in a basis randomly selected from the same set. The fact that quantum mechanics prohibits the exact copying of a photon ensures that any eavesdropper who intercepts, measures, and attempts to pass the photons on to the receiver will be unable to faithfully reproduce that signal. The presence of the eavesdropper can then be detected, prior to any exchange of information, by an examination of the error rate between portions of the keys generated by the sender and receiver.

Using a biphoton source, we have constructed a QKD system for use in research towards naval applications.

14. SUBJECT TERMS	**15. NUMBER OF PAGES**
secure communication, quantum key distribution, QKD, entanglement, Bennet-Brassard 1984 protocol, BB84, Eckert	123
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

QUANTUM KEY DISTRIBUTION LABORATORY DEMONSTRATION

Jack R. Brault
Lieutenant, United States Navy
BS, California State University - Bakersfield, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN PHYSICS

from the

NAVAL POSTGRADUATE SCHOOL
December 2021

Approved by: Frank A. Narducci
Advisor

Jihane Mimih
Second Reader

Joseph P. Hooper
Chair, Department of Physics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Quantum key distribution (QKD) is a method of secure key distribution which provides protection against the tampering and interception of information.

Following the Bennet-Brassard 1984 (BB84) protocol of QKD, we select randomly from a set of bases in which to produce polarized photons and send the photons to a receiver, who measures them in a basis randomly selected from the same set. The fact that quantum mechanics prohibits the exact copying of a photon ensures that any eavesdropper who intercepts, measures, and attempts to pass the photons on to the receiver will be unable to faithfully reproduce that signal. The presence of the eavesdropper can then be detected, prior to any exchange of information, by an examination of the error rate between portions of the keys generated by the sender and receiver.

Using a biphoton source, we have constructed a QKD system for use in research towards naval applications.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AOM acousto-optic modulator

BB84 Bennett-Brassard 1984

BS beam splitter

CAVR Center for Autonomous Vehicle Research

DARPA Defense Advanced Research Projects Agency

DH Diffie-Hellman key exchange

E91 Eckert 1991

EAM Emergency Action Message

EOM electro-optic modulator

FPGA field-programmable gate array

FSO free space optical

HAP High Altitude Platform

HOM Hong-Ou-Mandel

NC3 Nuclear Command, Control, and Communications

NCT No Cloning Theorem

NPS Naval Postgraduate School

NSA National Security Agency

PBS polarizing beam splitter

PKI public key infrastructure

QBER quantum bit error rate

QKD quantum key distribution

RF radio frequency

RSA Rivest-Shamir-Adleman key exchange

SPDC Spontaneous parametric down conversion

SKI secret key infrastructure

SPCM single photon counting module

UAV unmanned aerial vehicle

UUV unmanned underwater vehicle

Acknowledgments

I would like to thank the Navy for providing me with a graduate education, and for giving me the opportunity to spend the months of COVID-19 lockdown in beautiful Monterey, CA. I am very grateful to my advisor, Dr. Frank Narducci, for his patience in guiding me through the process of learning how to conduct experiments and adhere to scientific standards. I am thankful to Dr. Jeff Lee for his eagerness and willingness to help me in the lab. Special thanks go to Dr. Jihane Mimih, who provided me with a plethora of research material and also served as my second reader; CAPT Markus "Goody" Gudmundsson, who went out of his way to ensure that I would have the time I needed to finish my thesis; CDR Ana Tempone, who assisted me through some thorny administrative obstacles; and to my partner, Aleena, who always inspires me.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction/Motivation

Quantum technologies have the potential to provide important capabilities to the U.S. military in the near future. The most mature quantum technology, quantum key distribution (QKD), is poised to deliver previously-impossible capabilities in relation to secure communications and Nuclear Command, Control, and Communications (NC3).

This chapter is intended to provide the reader with a brief overview of QKD and the types of capabilities it may provide to the U.S. military. A more technical exploration of QKD is provided in Chapter 2. A detailed description of a QKD demonstration assembled at the Naval Postgraduate School (NPS) in Monterey, CA is provided in Chapter 3.

Much of the discussion around QKD is steeped in the language of cryptology. Appendix A is intended to act as a handy reference for readers who may not yet be acquainted with cryptology, including vocabulary and introductory-level explanations of fundamental concepts.

1.1 Overview of QKD

QKD is a secure key distribution infrastructure which allows for the secure sharing of one-time pads between remote locations. It works by encoding keys in quantum states, typically the states of individual photons, and then applying a transmission protocol to share those states between intended parties. Quantum principles prevent access by unintended parties to the key, and also alert the intended parties to the presence of an intruder. Quantum theory provides a theoretically perfect security basis for QKD.

Implementations of QKD can be classified either as *fiber-based* or *free space*. Fiber-based QKD uses optical fibers to transport photons between stations, while free space QKD uses laser communications.

A 2020 article by Uppal [1] discusses the military utility of laser communications, also known as free space optical (FSO) communications, currently under exploration by various research arms of the U.S. military. The advantages of FSO over conventional RF communications include greater bandwidth and lower signature, while the disadvantages include sensitivity to atmospheric conditions and difficulties associated with line-of-sight communications.

As long as two parties are able to conduct fiber-based or FSO communications with one another,

they can use QKD to constantly generate secure key material. Key material generated via QKD is in theory perfectly secure, and information encrypted via QKD is secure forever, regardless of the technological resources of any potential hacker or attacker[1] which they can later use to facilitate secure communications.

1.1.1 Capabilities

The pertinent capabilities of QKD include:

- Secure remote key sharing.
- Eavesdropper detection.
- Perfect security against eavesdroppers.

QKD is unique among key distribution infrastructures in its ability to provide all three of these capabilities at once.

1.1.2 Development

Quantum theory has been mature for decades, and many exciting technologies have been proposed to take advantage of various quantum principles. However, the difficulty associated with engineering, maintaining, and manipulating quantum states has made it difficult to develop these proposals into useful quantum technologies. Recently, technological advancement has begun to catch up with theory, and many quantum technologies are now tantalizingly close to realization. The simplicity and utility of QKD have propelled it to the forefront of the nascent quantum technology revolution.

Beginnings (pre-2004)

The first idea for using quantum states to facilitate secure communications began circulating in academia in the late 1960's. Called conjugate coding, after its use of conjugate quantum states for information exchange, it was originally conceived for the creation of unforgeable bank notes. Conjugate coding could not be implemented at the time due to the fragility of quantum states, and it remained only a conversation piece until further theoretical breakthroughs warranted its publication over a decade later (Wiesner, 1983 [2]). Bennet and Brassard published the first paper on QKD [3] in 1984. They were also the first to demonstrate QKD in a laboratory environment [4] in late 1989. This demonstration ignited interest in several scientific communities, leading to an alternate theoretical formulation by Ekert in 1991 [5] and laboratory demonstrations using various protocols throughout the 1990s.

[1] This condition is also called "information-theoretical security" See Appendix A for more information.

Development (2004–2017)

Christandl, Renner, and Ekert presented a generic security proof [6] for QKD in 2004. The first large-scale test of QKD was then conducted at the Defense Advanced Research Projects Agency (DARPA) Quantum Network Testbed (Elliot, 2007 [7]) in Massachusetts. The test ran from 2004 to 2007, testing many features, protocols, and permutations of QKD. In the following years, a number of QKD networks were built for academic use across the world (EU in 2008 [8]; China in 2009 [9]; Japan in 2010 [10]).

Application (2017–present)

In 2017, free space QKD was demonstrated between ground and satellite stations by a team of Chinese researchers (Yin et al., 2017 [11]; Liao et al., 2018 [12]) and between ground stations and moving aircraft by a team of Canadian researchers (Pugh et al., 2017 [13]). More recently, QKD has found favor among institutions and businesses which manage large data centers. The 2019 *Inside Quantum Technology Report* predicts that QKD will be a nearly billion-dollar industry by 2024 (Gasman, 2019 [14]). As of early 2021, the Indian Space Research Organization has announced successful demonstrations of QKD and intentions to develop a satellite-based QKD network (Indian Space Research Organization, 2021 [15].)

1.1.3 Challenges

The conversation around QKD implementation in the United States has been contentious. Its cost, convenience, and ability to deliver on its promise of security have all been subject to debate and scrutiny. The most conspicuous public discussion of QKD's shortcomings came in a statement released by the National Security Agency (NSA) in 2020 [16].

NSA statement on QKD

In the statement, the NSA officially advised against using QKD as a primary security strategy. The statement warns that QKD is not yet a mature technology, and that its implementation might require massive investment in infrastructure while only providing minimal short-term security benefits. The NSA also warns about issues concerning user authentication and Denial of Service attacks.[2]

The NSA's strongest case against QKD is that QKD requires the fielding of hardware which has not yet been shown to be as secure as the equipment used in conventional key distribution. Until this problem can be resolved, the NSA has chosen to work alongside the National Institute of Standards and Technology to develop classical quantum-proof cryptography systems.

[2]These claims are contested by QKD industry representatives familiar with the technology (Prisco, 2020 [17]).

This public stance from the NSA has set the tone for the U.S. national defense sector, leaving the burden of QKD development to academic groups, government agencies, and industry.

Overcoming challenges

Despite the NSA's stance, the U.S. Department of Energy has decided to invest millions in QKD infrastructure and quantum technology research (Department of Energy, 2019 [18]). In addition, tech companies, especially in the telecommunications industry, have already begun developing QKD infrastructure in an effort to future-proof their networks (Verizon, 2020 [19]) (see Appendix A for more information regarding future-proofing.)

Currently, the largest hurdle to widespread adoption of QKD is range. Quantum states are fragile, so nodes in fiber-based QKD networks must be relatively close to one another (less than 1000km). To address this issue, research and development efforts are currently concentrated on the development of a device known as a *quantum repeater*. A quantum repeater is a device which would enable one to create long-range entanglement between two distinct and far locations by dividing the transmission distance into short segments, creating entanglement between each of these segments, and expanding entanglement distance between segments using entanglement swapping. Until such a device is developed, fiber-based applications of QKD will be limited. Similarly, free space applications of QKD are hampered by *scintillation*, the random optical power loss due to atmospheric turbulence. However, the 2017 satellite-to-ground and ground-to-aircraft QKD demonstrations mentioned in Section 1.1.2 show that free space QKD is capable of operating at ranges greater than 1000km, which is sufficient for many military applications.

1.2 Potential Military Applications of QKD

Since the 2004-2007 DARPA QKD testing discussed in [7], investigations into military applications of QKD in the U.S. have been carried out mostly by industry specialists (Russel, 2008 [20]). Recent advances in QKD could increase reliability of military key distribution systems and provide critical communication capabilities (Hall and Sands, 2020 [21]).

1.2.1 Continuity of Operations

A QKD-capable deployed unit (such as a satellite, submarine, high altitude aircraft, special operations team, etc.) would be able to recover from expiration, loss, battle damage, theft, sabotage, or any other loss of its cryptological material, without ever leaving the field and without any transfer of physical media. In addition, the small signature of FSO allows QKD to be conducted with a very low risk of counter detection, so secure key material could be generated passively throughout

a unit's deployment as long as line of sight was maintained. Alternatively, line of sight communications could be established situationally in order to assist the deployed unit in recovering from an unplanned loss of cryptological material.

1.2.2 Submarine Nuclear Command, Control, and Communications (NC3)

There are many possible military applications of QKD. The specific example of submarine NC3 is discussed below.

Submarine NC3 background

Current U.S. submarine NC3 infrastructure involves the transmission of an Emergency Action Message (EAM)—for example, an order to launch a nuclear missile at a specified target—from decision makers to deployed Ballistic Missile Submarines (SSBNs) via VLF signals relayed from ground-based transmission sites or deployed aircraft. EAMs are received onboard the SSBN in the form of compact strings of typewritten phonetic-alphabet characters, which crewmembers must error-correct and decipher with the assistance of an array of physical instruction manuals and code books. EAM processing must be done by hand because current submarine key distribution infrastructure is based on secret key infrastructure (SKI), which, as described in Appendix A, is demonstrably secure but relies on physical media for key distribution. The system is designed to be both *robust* and *risk-averse* .

Robust design

The VLF communication spectrum was chosen for submarine NC3 because it is the most robust of the available methods of communication with a submarine in an alert posture (that is, a submarine that is trying to remain hidden but still needs to be able to receive EAMs.) The drawback of VLF is that EAM transmission via VLF takes a long time, on the order of minutes, which allows environmental factors (such as wave action, or ionization of the atmosphere due to nuclear detonations) to interfere with the communication link between the transmitting platform and the submarine during EAM transmission. This can occasionally lead to high error rates and incomplete transmissions. Even so, the system is designed so that a well-trained crew can process an EAM in a reasonable amount of time even in an environment where communication systems are significantly degraded.

Risk-averse design

The probability that human error during EAM processing could lead a crew to take incorrect actions with nuclear arms (such as launching a missile at an incorrect target) is minimal. However, there is a very real chance that human error might cause a significant time delay in carrying out the orders

contained in the EAM. Specifically, there is a phase of submarine NC3 where an encrypted EAM is received onboard but its contents are not yet known to the crew. The processing-by-hand system means that, depending on the readability of the message and the proficiency of the crew, this period might last anywhere between a few seconds and a few minutes. In the event that human error is present, this phase might last significantly longer. This possibility is problematic because there is a specific planned time window between the time of receipt of an EAM and the execution of the orders. If human error causes a delay in the onboard processing, this time window can be missed, resulting in a failure to achieve critical mission objectives.

Application of QKD

QKD offers a way to reduce the probability of human error, and thus the risk of a significant time delay, without sacrificing the robustness or risk-averse nature of submarine NC3. Instead of the current SKI-based method of loading physical code books while in port, a QKD-capable submarine could coordinate with another QKD-capable platform to generate secure key material.[3] The keys thus generated could later be used to encrypt EAMs transmitted to the submarine, and, since QKD does not rely on physical media, onboard EAM processing could be conducted electronically. This would allow the crew to begin taking action in accordance with the contents of an EAM immediately, effectively bypassing the phase where human error might cause the most significant delays.

[3]Note that this could be done in port with ground-based platforms, or at sea with airborne platforms such as an aircraft, satellite, or High Altitude Platform (HAP), while the submarine was in a condition which allowed for active communications.

CHAPTER 2:
QKD Theory vs. Reality

Chapter 2 briefly acquaints the reader with the theory and mechanisms of QKD. Section 2.1 presents the basic underlying theory, and Section 2.2 discusses feasibility and the difficulties associated with real world implementation. This discussion is intended to be introductory in nature, with an emphasis on the physics of QKD, as opposed to the complex cryptological aspects. For a more thorough review of the current state of QKD theory, implementation challenges, practical security, and feasibility, the reader is directed to Xu, et al.'s 2020 review article "Secure Quantum Key Distribution with Realistic Devices" [22], published in the journal Reviews of Modern Physics.

2.1 Theory

As discussed in Chapter 1, the theory behind QKD was developed and published by Bennett and Brassard. Their 1984 paper presents a novel (at the time) protocol for the distribution of information-theoretic secure one-time pads (keys), at range and with security from man-in-the-middle attacks guaranteed by quantum principles. This section describes the fundamentals of how to distribute secure keys using quantum states.

2.1.1 Basic Theory

At its most basic, QKD is simply the act of encoding keys in quantum states and distributing them as desired. By carefully controlling the method of distribution, one can exploit certain quantum principles in order to provide security to the process, thereby guaranteeing that a key is known only to the desired parties.

Basic protocol

Methods of key distribution are called protocols. A protocol typically consists of two phases: **key generation** and **key processing**.

- **Key Generation Phase.** This phase consists of the actual distribution of the key from an operator at point A, traditionally called Alice, to an operator at point B, called Bob. For free space QKD, this phase is conducted via FSO.
- **Key Processing Phase.** In this phase, Alice and Bob take any necessary steps to ensure that the key generation phase was successful, and that there were no eavesdroppers. This phase may be conducted over any communication channel available to Alice and Bob regardless of

the security of that channel, since the activities performed during key processing do not risk the exposure of any secure information.

Key generation phase

Key generation requires Alice to prepare a quantum state and transmit it, and then for Bob to detect and record the state. They repeat these steps until the desired key length is achieved. For example, the quantum state used to encode a one-time pad could be the polarization angles of individual photons. Alice and Bob might make an arrangement that when Alice transmits a single photon with polarization angle $\phi_{signal} = \alpha$ to Bob, he should record the event as a binary 0, and when she transmits a photon with polarization angle $\phi_{signal} = \alpha + \pi/2$ he should record that event as a binary 1.

In order to prepare the quantum state, Alice first needs a device for generating single photons. Ideally, she would have a source that could generate single photons on demand, but such a source does not exist. More likely, she will have a source which produces single photons randomly according to some known distribution, and she will utilize a fast shutter so that it is very unlikely that more than one photon will be transmitted per cycle of the shutter. Once she has calibrated her shutter so that she is reliably controlling her photon signal, Alice can rotate the polarization angles of the photons and transmit them to Bob.

In order to measure the polarization angles of incident photons, Bob may install a polarizing beam splitter (PBS), aligned to axis $\phi_{PBS} = \beta$, so that incident light with a polarization angle aligned along this axis will be transmitted to one output path and incident light polarized along an orthogonal axis will be reflected to another. If incident photons are aligned to a polarization angle $\phi_{signal} = \alpha$, then there is a probability $\cos^2(\alpha - \beta)$ that the PBS will transmit the incident photon, and a probability $\sin^2(\alpha - \beta)$ the PBS will reflect it. Bob can thus determine the polarization angle of an incident photon by placing detectors at each output of his properly aligned PBS and noting which detector detects the photon. This method of polarization angle detection works only as long as the PBS is aligned to the polarization basis of the incident photons, so that $\beta = \alpha + n\pi/2$, where n is a whole number. If the offset between α and β is not a half-integer multiple of π, then any transmitted photon will have some non-zero probability of ending up at the wrong detector.

After the key generation phase is complete, Alice has the list of states that she transmitted and Bob has a list of what his detection apparatus detected for the transmission events corresponding to each index on Alice's list. These respective lists are known as their "raw" keys. The raw keys are the final products of the key generation phase, and they are the inputs to the key processing phase.

Key processing phase
The key processing phase can be subdivided into the **sifting step** and the **checking step**.

- **Sifting step.** The sifting phase consists of whatever actions Alice and Bob take to ensure that their keys are as similar as possible. For example, during the sifting phase Bob will examine his raw key and tell Alice the index of any event where he detected a total of either zero or more than one photon, and they will both discard that bit index. After the sifting phase is complete, their raw keys are known as "sifted" keys.
- **Checking step.** The sifted keys are then checked for security in the checking phase. The checking phases consists of Alice and Bob choosing some subset of their sifted keys and revealing them to one another over the chosen non-secure communication channel. As will be discussed shortly, simply comparing a subset of their sifted keys and calculating the error rate between the two is sufficient to reveal if the transmission was compromised.

After the processing phase, Alice and Bob each have possession of an information-theoretically secure one-time pad, which can be used to exchange secure communications whenever they desire. Ostensibly, Alice and Bob can use this simple procedure to generate key material as quickly as Alice can generate and transmit the polarized photons.

2.1.2 "Man in the Middle" Attacks

Suppose then that an aspiring eavesdropper, Eve, decides to carry out an attack against such a system by intercepting the photons en route from Alice to Bob (this is called a "man in the middle" attack, despite its characterization in the literature as being perpetrated by Eve, who canonically uses she/her pronouns.) Eve has two options for her attack:

- **Passive eavesdropping.** Eve might attempt to passively amplify the transmitted signal. That is, she might try to put some kind of quantum cloning device in between Alice and Bob, and allow it to clone each transmitted photon without disturbing its quantum state. The original photon would then continue on its journey to have its polarization angle be measured by Bob, while Eve could measure the polarization angle of only the cloned photon.
- **Active eavesdropping.** Assuming Eve knows the polarization angle basis which Alice is using, Eve can simply measure the polarization angle of the transmitted photons, then make a copy and send the copy on to Bob.

The strength of QKD is that it allows Alice and Bob to defend against both of these attacks by leveraging quantum principles. The passive eavesdropping attack is addressed by the No Cloning Theorem, which is discussed in Section 2.1.2, while the active attack is addressed by QKD protocols,

which are discussed in Section 2.1.2.

Defense against passive eavesdropping: the No Cloning Theorem

The theoretical security of QKD systems against passive eavesdropping is based in the No Cloning Theorem. The No Cloning Theorem is stated succinctly by Wootters and Zurek in the title of their 1982 paper, "A single quantum cannot be cloned" [23]. We now follow the logic of their proof and determine the veracity of this statement.

The proof describes a *gedanken* experiment wherin an arbitrary quantum state s is subject to the effects of an alleged quantum cloning device $\hat{A}$. For demonstration purposes, it is assumed that s is in the rectilinear basis, such that $|s\rangle = \alpha |s\rangle_H + \beta |s\rangle_V$, and therefore

$$\begin{aligned} \hat{A} |\psi_i\rangle |s\rangle |0\rangle &= |\psi_f\rangle |s\rangle |s\rangle \\ &= |\psi_f\rangle [\alpha^2 |s\rangle_H |s\rangle_H + \beta^2 |s\rangle_V |s\rangle_V + \alpha\beta(|s\rangle_H |s\rangle_V + |s\rangle_V |s\rangle_H)]. \end{aligned} \tag{2.1}$$

We then examine the behavior of the cloning device on a state s oriented entirely along one of its basis vectors, $|s\rangle = |s\rangle_H$ or $|s\rangle = |s\rangle_V$, and find that

$$\hat{A} |\psi_i^H\rangle |s\rangle_H |0\rangle_H = |\psi_f^H\rangle |s\rangle_H |s\rangle_H \tag{2.2}$$

or

$$\hat{A} |\psi_i^V\rangle |s\rangle_V |0\rangle_V = |\psi_f^V\rangle |s\rangle_V |s\rangle_V . \tag{2.3}$$

If, instead, s were a superposition of the basis vectors, $|s\rangle = \alpha |s\rangle_H + \beta |s\rangle_V$, then the final state after interacting with the cloning device should simply be the superposition of the states described in Equations 2.2 and 2.3, so that

$$\hat{A} |\psi_i\rangle |s\rangle = \alpha\psi_f^H |s\rangle_H |s\rangle_H + \beta\psi_f^V |s\rangle_V |s\rangle_V . \tag{2.4}$$

Equation 2.4 contradicts Equation 2.1, and thus it is shown that the arbitrary quantum state s cannot be cloned. Returning to the original hypothesis— "a single quantum cannot be cloned"—we find it somewhat lacking: in the special case where one of the weighting factors, say α, were set equal to unity, while β were set equal to zero, and imposing the condition that the cloning device must be left in a pure state ($|\psi_f^H\rangle = |\psi_f^V\rangle$), there is no contradiction: the cloning device can theoretically operate under these conditions; that is, it is theoretically possible to amplify a *known* quantum state. The No Cloning Theorem can be revised to state, "an *arbitrary* quantum cannot be cloned."

With this axiom in place, passive eavesdropping of a QKD system is theoretically defeated by quantum principles. Revisiting the QKD system described in Section 2.1.1, where the quantum state employed is the polarization angle of single photons, we find that if Eve calibrates her quantum cloning device to clone photons with a polarization angle of α, then any time she intercepts a photon with a polarization angle of $\alpha + \pi/2$ the device will fail: both the original photon and the cloned photon will be left in a superposition state, so that the polarization angle no longer holds any information. When this happens, the detector at Bob's station which ends up detecting the photon will have no correlation to the state of the state of the photon originally prepared by Alice.

Thus, when measured by Bob, the cloned photons will have an equal chance of being measured as a binary 0 or as a binary 1. During the key processing phase, specifically the checking phase, Alice and Bob need only compare a portion of their key material and they will find that any time Alice sent a 1 there is a 50% chance that Bob will have recorded a 0 (overall this corresponds to an eavesdropper-induced error rate of 25%.) This will alert them to Eve's presence. They will then have a chance to address the attack on their communications and try again, and at no point will their sensitive communications be compromised.

Defense against active eavesdropping: BB84 Protocol

Theoretical security against active eavesdropping requires an active QKD protocol. In 1984, Bennett and Brassard published the first secure QKD protocol, aptly named Bennett-Brassard 1984 (BB84), describing a straightforward method to thwart any man-in-the-middle attacks against QKD systems by using some basic principles of photon polarization states. A handful of other protocols have been proposed since, utilizing different quantum states to optimize for various use cases—still, the simplicity and generality of BB84 make it ideal for demonstration purposes, and thus it will be the one discussed in this paper.

We begin by defining two polarization bases relative to some angle $\phi = \beta$: the rectilinear basis, composed of the basis vectors $(1, 0)$ and $(0, 1)$ ($\phi = \beta, \phi = \beta + \pi/2$) and the diagonal basis, composed of basis vectors $(\sqrt{2}/2, \sqrt{2}/2)$ and $(-\sqrt{2}/2, \sqrt{2}/2)$ ($\phi = \beta + \pi/4, \phi = \beta + 3\pi/4$). Suppose

that for each photon Alice transmits, she randomly chooses either the rectilinear or diagonal basis in which to send it. In addition, suppose that for each photon transmission Bob randomly chooses to align his PBS either to angle $\phi_{PBS} = \beta$ or $\phi_{PBS} = \beta + \pi/4$. By these two actions, Alice and Bob effectively randomize the basis in which the key material is being generated for each bit.

In order to intercept and successfully measure the polarization angle of the photons Alice transmits, Eve must have her own PBS aligned to the correct basis—and, assuming that Alice and Bob's stations are secure (Eve only attacks the space in the middle), her efforts are reduced to guesswork. Each time she guesses wrong, her measurement (and thus her re-transmission) has a 50% chance of being wrong, for a total eavesdropper-induced error rate of 25%. Again, Alice and Bob can easily detect Eve during the key processing phase, and their sensitive communications will remain secure.

Note that Alice and Bob each choose their bases randomly and independently, and thus they will choose differently half the time. BB84 protocol accounts for this during the sifting phase. The two of them share which basis they chose for each bit index, and they discard any bit corresponding to an instance where they chose different bases.

2.2 Reality

The combination of the No Cloning Theorem and a secure protocol provides the theoretical basis for the security of QKD. After the key processing phase, if no eavesdropper was detected, Alice and Bob will both be in possession of an information-theoretic secure one time pad which is every bit as secure as a key distributed using SKI but which requires no exchange of physical media, and can be generated at range. It should come as no surprise, however, that when theory meets reality, some of the nice theoretical results of QKD begin to tarnish.

Some major criticisms of QKD are that it generates key material too slowly in real world implementations, and that the perfect security promised by theory cannot actually be realized. Here, we analyze the feasibility of free-space QKD in a real world environment. Sections 2.2.1 through 2.2.3 discuss various applicable system parameters, technological limitations, and environmental variables, and explore their effects on the security and rate of generation of key material. Section 2.2.4 compares the results of these discussions to real-world needs and shows that QKD is capable of performing necessary tasks using currently available technology.

The following analysis is written at an accessible level in order to demonstrate that free space QKD can be conducted at useful speeds. For a more rigorous analysis of the theoretical speeds and security levels achievable by real QKD systems, the reader is referred to [22].

2.2.1 Transmission Speed Overview

We begin our analysis by defining the relationship

$$B = L_k/T, \tag{2.5}$$

where B is the rate of generation of secure key material, or the "bit rate" L_k is the total number of bits in a secure key, or "secure key length" and T is the total time required to generate a secure key of length L_k.

Each "transmission event"—for example, a cycling of the fast shutter described in Section 2.1.1—results in the generation of one raw key bit for each participant. The total number of raw key bits is referred to as L_r, the raw key length. Each raw key bit has a probability P of becoming a bit of the final key held by the sender and receiver,

$$L_k = L_r \times P. \tag{2.6}$$

The total time T required for key transmission is the product of the total number of transmission events L_r and the time required to transmit one raw key bit, t_{bit}, so that

$$T = L_r \times t_{bit}. \tag{2.7}$$

Substituting Equations 2.6 and 2.7 into Equation 2.5, the key length cancels and the bit rate B is expressed as

$$B = P/t_{bit}. \tag{2.8}$$

2.2.2 Transmission Probability P

The value of P is determined by the probabilities of the possible loss mechanisms for each raw key bit, of which there are two: sifting and error rate checking. The probability that a raw key bit is not discarded via sifting is P_{ns}, and the probability that a raw key bit is not included in the error rate checking procedure is P_{nc}, so that

$$P = P_{ns} \times P_{nc}. \tag{2.9}$$

Loss mechanism 1: Sifting factor P_{ns}

The value of P_{ns} is determined by the raw key bit loss mechanisms present in the sifting phase, which include basis matching P_{bm} and bit logic P_1.

$$P_{ns} = P_{bm} \times P_1. \tag{2.10}$$

Basis matching factor P_{bm}

Basis matching refers to the probability P_{bm} that the sending basis and receiving basis for any given transmission event will match. The sender and receiver each choose from one of two bases randomly for each event, so the probability of a basis match is one half:

$$P_{bm} = 1/2. \tag{2.11}$$

Bit logic factor P_1

Bit logic refers to the probability P_1 that a given transmission event will result in the detection of exactly one photon by the receiver. If a given photon source follows a Poissonian distribution, for example, P_1 will also have a Poissonian distribution with a mean $\lambda_{det} = N \times P_{na}$, where N represents the mean number of photons transmitted per transmission event and P_{na} is the optical non-attenuation factor. N is calculated from the rate of photon generation at the source, λ_{src}, and the length of each transmission event τ_w, such that $N = \lambda_{src} \times \tau_w$ and therefore $\lambda_{det} = \lambda_{src}\tau_w P_{na}$. Poissonian statistics dictate that the probability of any particular value n being realized is $P_n = \frac{\lambda^n}{n!} \exp(-\lambda)$. Making all necessary substitutions, P_1 is found to be

$$P_1 = \lambda_{det} \exp(-\lambda_{det}) \tag{2.12}$$

for a Poissonian source with a mean of

$$\lambda_{det} = \lambda_{src}\tau_w P_{na}. \tag{2.13}$$

Non-attenuation factor P_{na}
Non-attenuation factor P_{na} is the probability that any individual photon transmitted by Station A will be detected at Station B. The value of P_{na} is highly dependent on individual applications. Photon attenuation mechanisms can be subdivided into two broad categories: apparatus-specific attenuation and environmental attenuation.

Apparatus-specific attenuation is attenuation due to optical components in the apparatus itself. For well-aligned and calibrated optical systems operating in benign environmental conditions, transmitting photons via currently available adaptive optics technology, the major loss mechanisms will be the imperfect efficiency of Bob's detectors and the optical transmission signal telescopes installed at both Station A and Station B.

Detector efficiency varies widely by model. A Si-APD detector, suitable for free space optics (500nm−900nm range), might have a maximum efficiency of around $0.4 - 0.7$ (Excelitas SPCM-AQRH family data sheet [24]). An In-GaAs/InP detector, more suitable for fiberized telecommunications (1100nm-1700nm range), might have an efficiency of around $0.1 - 0.25$ (MPD In-GaAs/InP single photon detector data sheet [25]). More advanced detectors which use superconducting materials might have an efficiency of over 0.9 for a wide range of wavelengths, but the cost and engineering requirements might be prohibitive (these detectors require sub-1K cryogenics) (Single Quantum SNSPD data sheet [26]. For the purposes of this discussion, we will assume that Alice is transmitting 810nm photons over free space, and that Bob is detecting them using a Si-APD detector (efficiency ~ 0.6 at 810nm).

Telescopes used for beam collimation also come with a variety of efficiencies, but tend not to be much worse than 0.9. Assuming that Alice and Bob each use a telescope with an efficiency of 0.9, the overall system non-attenuation factor can be estimated as

$$P_{na}(sys) \approx 0.6 \times 0.9 \times 0.9 \approx 0.5. \tag{2.14}$$

Environmental attenuation refers to attenuation caused by the medium through which Alice's signal is transmitted. For free space QKD, that medium is the atmosphere. Environmental attenuation is negligible over short distances, such as distances within a lab, or in benign environmental conditions. As the distance from Station A to Station B increases, attenuation due to atmospheric turbulence and scintillation may increase rapidly, especially in adverse weather conditions. Environmental attenuation is highly variable and difficult to account for in general, but it does not preclude long-range free space QKD—in fact, in 2017 Yin et al. demonstrated free space QKD from a satellite in low-earth-orbit to ground [11]. For the remainder of this discussion, it will be assumed that the key distribution is being performed under benign conditions, and so the environmental attenuation factor is negligible.

Loss mechanism 2: Checking factor P_{nc}

Each transmission event which does not get sifted out is recorded by the receiver as one bit of the sifted key, which has a length $L_s = L_r \times P_{ns} = L_k/P_{nc}$. The total number of bits in the sifted key which are subsequently used during the checking phase is given by $L_c = L_s \times (1 - P_{nc})$, so that $L_c = L_k \frac{1-P_{nc}}{P_{nc}}$. Rearranging to solve for P_{nc},

$$P_{nc} = \frac{1}{1 + \frac{L_c}{L_k}}. \tag{2.15}$$

The value of P_{nc} is chosen so that, in the absence of an eavesdropper, the error rate of the checked bits should lie within the $(1 - \alpha)\%$ confidence interval of the system error rate, or

$$(1 - \alpha)\%CI = [\mu \times (1 \pm \alpha)] = \mu \pm z_{\alpha/2}\frac{U_{sys}}{\sqrt{L_c}}, \tag{2.16}$$

where μ is the mean system error rate and $z_{\alpha/2}$ is the critical value for the chosen value of α. Rearranging Equation 2.16 gives the minimum value of L_c:

$$L_c = (CV_{sys} \times \frac{z_{\alpha/2}}{\alpha})^2, \tag{2.17}$$

where the coefficient of variation $CV_{sys} = U_{sys}/\mu$ is a system constant which is typically small (for the QKD demonstration apparatus described in Section 3.1, CV_{sys} was measured to be approximately

one half). Assuming a restrictive value for α of 0.01, corresponding to a 99% confidence interval, L_c is found to be a constant with a magnitude on the order of a few hundred bits. This number is very small when compared to the length of a typical key, which can be many thousands of bits long. Assuming a key length L_k of at least ten thousand characters, the ratio $L_c/L_k << 1$ and Equation 2.15 can be approximated to zeroth order, so that

$$P_{nc} \approx 1. \tag{2.18}$$

Combining the values obtained for P_{bm}, P_1, and P_{nc} in Equations 2.11, 2.12, and 2.15 into Equation 2.9 gives the value of P:

$$P \approx \frac{\lambda_{det}}{2e^{\lambda_{det}}}. \tag{2.19}$$

2.2.3 Single-bit Transmission Time t_{bit}

Single bit transmission time t_{bit} is the length of time required for the sender to transmit one bit of the raw key. It is calculated as the sum of the time spent transmitting, τ_w, and the time spent preparing to transmit, τ_p:

$$t_{bit} = \tau_p + \tau_w. \tag{2.20}$$

The length of τ_w and τ_p are determined by technological limitations. Figure 2.1 shows a representation of the construction of one time bin t_{bit}.

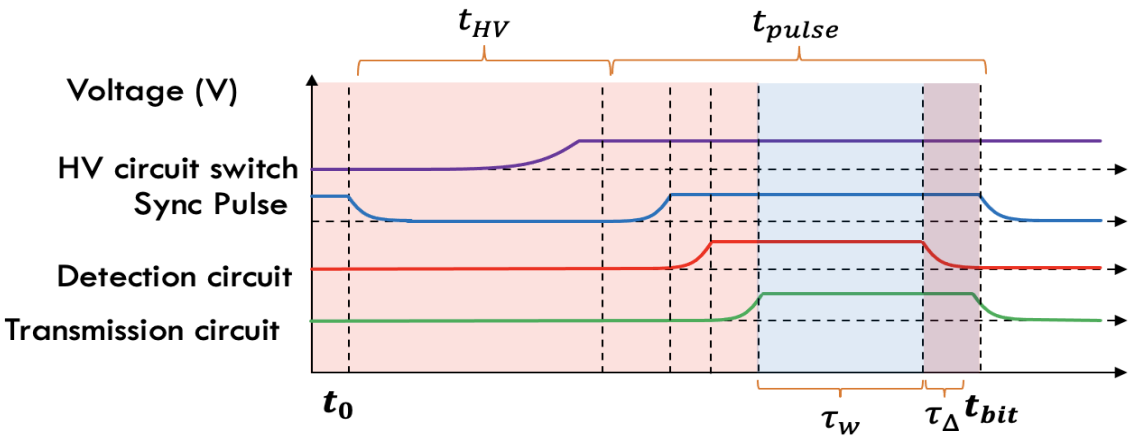

Figure 2.1. Representative time diagram of a single bit transmission. Transmission window preparation time τ_p is indicated by the red-shaded area. Transmission window τ_w is indicated by the blue-shaded area. The purple-shaded area indicates τ_Δ, the desync factor between Stations A and B, which ultimately sets the lower limit for the length of τ_w.

Transmission window preparation time τ_p

The transmission preparation time τ_p consists of the "dead" time in between transmissions. It includes the time required for electronic commands to travel throughout the apparatus, the rise and fall times of the HV signals applied to the electro-optic modulator (EOM)s, the opening and closing of the transmission window, and any buffers put in place to account for jittering. As shown in Figure 2.1, the dominant term in τ_p is t_{HV}, the time required to switch the HV circuit. While the HV rise time can be on the order of just a few tens of nanoseconds, the circuitry itself usually requires more time to stabilize and prepare for another switch, and as such the best commercial/off-the-shelf (COTS) HV switching circuits are limited to about 100kHz for extended operations, corresponding to a t_{HV} length of about 10µs. Because this term is so large compared to the other factors, $\tau_p \approx t_{HV}$, and so

$$\tau_p \approx 10\mu s. \tag{2.21}$$

Transmission window τ_w

It is impossible to know exactly when a photon source will generate a single photon. Photon sources generate a "signal" made up of randomly generated photons following a particular distribution. Assuming a source whose distribution is Poissonian, a single photon can be picked out of a randomly generated signal by employing a fast shutter, which opens for a set length of time τ_w. During that time, an average of $\langle N \rangle = \lambda_{src} \times \tau_w$ photons are transmitted, where λ_{src} is the mean

count rate of the source's distribution.

Information Insecurity Factor I

Setting the length of τ_w so that N is greater than one opens up a loophole to QKD's theoretical security. If N is equal to one hundred, for example, so that an average of one hundred photons were transmitted per event, Eve could choose to intercept only one of them. In that event, it is unlikely that she would ever be detected. Since the probability that more than one photon will be generated in a given transmission event can never be reduced to zero, this phenomenon introduces an information insecurity factor, I, which must be accounted for by any real QKD system.

In order to determine the value of I for a given QKD system, we make the following assumptions about Eve's attack:

1. Eve executes a man-in-middle attack by intercepting a certain number N_E photons out of the N transmitted photons per transmission event. Thus, for any transmission event attacked by Eve and subsequently *not* discarded during key processing, the probability that the single photon which is detected and retained by the receiver is actually a copy injected by Eve is the ratio of intercepted photons to transmitted photons N_E/N.
2. Eve executes the attack on a certain proportion of the total number of transmission events (this proportion is equal to the information insecurity factor, I.) The ratio of intercepted bits which make it into the final key to non-intercepted bits is then defined as $I \times N_E/N$.
3. When Eve intercepts a photon, she attempts to gain information from it utilizing either the passive eavesdropping attack or active eavesdropping attack described in Section 2.1.1—that is, she replaces any photon she intercepts.[4] Thus, the eavesdropper-induced error rate in the final key is the probability that any given intercepted photon will be left in an erroneous state, P_E, multiplied by the proportion of the final key that represents intercepted bits: $E = P_E I N_E/N$, where E is the eavesdropper-induced error rate.
4. The error rate E introduced by the attack is sufficiently small so that it is not detectable via the checking algorithm described in Section 3.1.3, or $E \leq U_{sys}$.

These assumptions lead to the following relationship between the system error rate uncertainty, U_{sys}, and the maximum information insecurity factor, I_{max}:

[4]This assumption might seem contrived at first, since it would be far easier for Eve to simply remove a photon without replacement. However, any real QKD system will be calibrated so that Bob will not normally receive more than one photon per transmission event: if he did, that bit index would be eliminated during the sifting phase. Therefore, if Eve makes a habit of removing one photon from the transmission without replacing it, Bob will likely not receive any photons for that event—again, the affected indices will be eliminated during the sifting phase. Eve only stands a good chance of gaining useful information if she replaces the photons that she intercepts, even if her replacements have a high error rate.

$$U_{sys} = I_{max}\frac{P_E N_E}{N}. \tag{2.22}$$

The value of P_E is already known from Section 2.1 to be $1/4$. For conciseness of notation, the value of N_E is set to one (although it can be higher.) Substituting these values into Equation 2.22, along with the definition $N = \lambda_{gen} \times \tau_w$, we find a linear relationship between τ_w and I_{max},

$$\tau_w = \frac{I_{max}}{4\lambda_{src}U_{sys}}. \tag{2.23}$$

Therefore, information security is maximized by minimizing τ_w.

Photon source distribution

Before suggesting that τ_w should be set as small as possible, we should examine the possibility that a very short transmission window will result in a large proportion of empty transmissions, or "empty bins." High proportions of empty bins means a low transmission efficiency, which in turn could hypothetically cause a slowdown in the rate of key material generation. This phenomenon is visualized in Figure 2.2, which shows the average number of photons produced during a given length of time τ for a photon source which follows a Poissonian distribution with a mean of 10^6 photons per second.

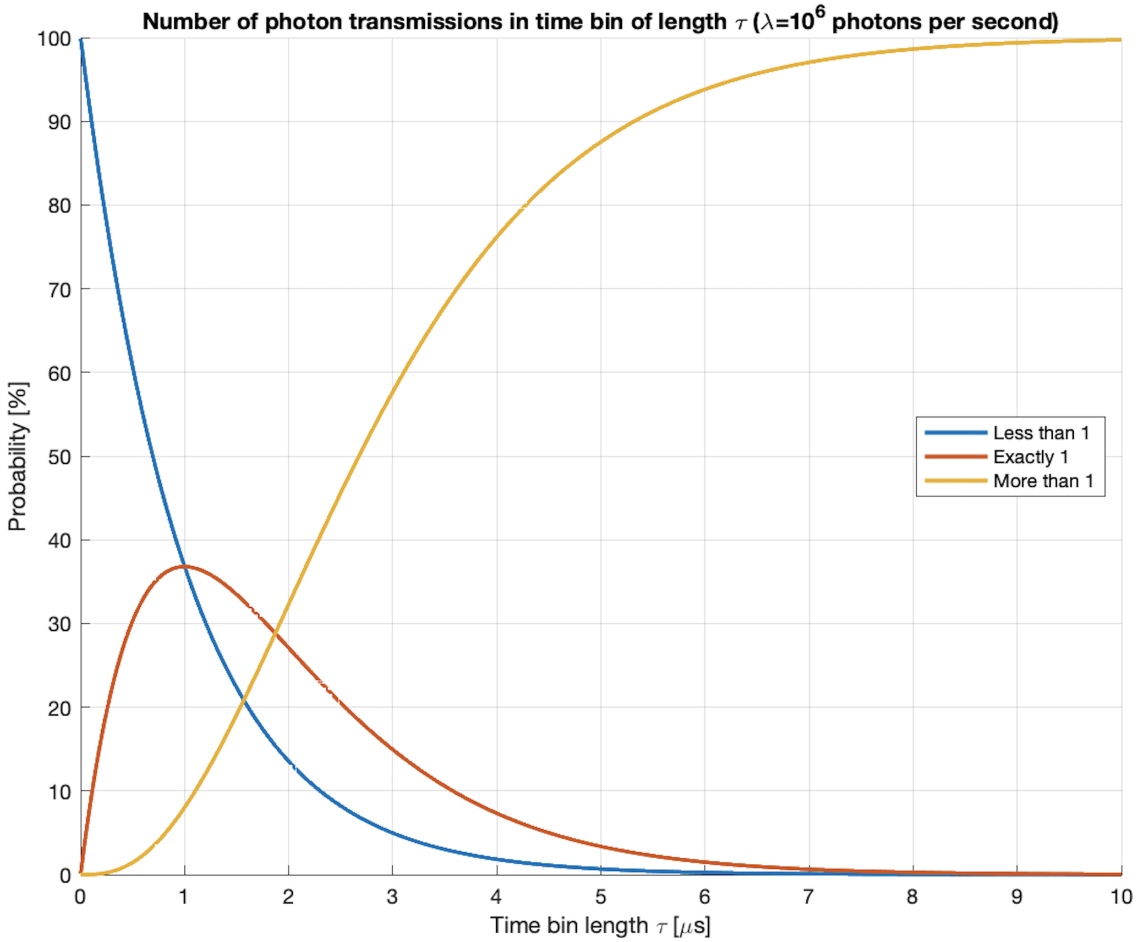

Figure 2.2. Average number of photons produced during a given length of time τ for a photon source which follows a Poissonian distribution with mean 10^6 photons per second.

The probability of generating *exactly one* photon (the red line in Figure 2.2) is maximized when $\tau \approx 1\mu s$. It can be seen in the figure that when τ is less than $1\mu s$, the rate at which the probability of generating *more than one* photon (the yellow line) follows a charging exponential shape, while the probability of generating *less than one* photon (the blue line) is approximately linear. In other words, as the transmission window shortens, the probability that more than one photon will be transmitted during the window—corresponding to information insecurity—*decreases* more quickly than the probability that no photons will be transmitted—corresponding to "empty bin" transmission inefficiency—*increases*. Thus, as τ_w is made arbitrarily small, information security improves faster than transmission efficiency deteriorates, indicating that τ_w really should be set as small as possible.

Desync factor τ_Δ

For a real system, there is a technological limit to how short τ_w can be. Alice and Bob must be able to synchronize their stations so that Bob's detection window aligns with Alice's transmission window (see Figure 2.1). The degree to which the two stations are not synchronized, denoted by the desync factor τ_Δ, determines the lower limit of τ_w. In order to maximize the probability that Alice will transmit only while Bob's detection window is open, it is necessary that $\tau_w \gg \tau_\Delta$ (the power of the photon source should be tuned accordingly, so that the desired value of N is achieved while keeping τ_w large compared to τ_Δ.)

For applications where Alice and Bob the synchronization must be done wirelessly, the level of synchronization required for QKD can be difficult to achieve. Recent work in the field of wireless sub-microsecond synchronization by Zengen et al. [27] has resulted in the possibility of a τ_Δ of $0.19\mu s$. Assuming this number is the current technological barrier, and imposing the condition that τ_w must be greater by at least an order of magnitude, we can estimate the current minimum transmission window length τ_w as

$$\tau_w \approx 2\mu s. \tag{2.24}$$

Substituting the values for τ_p and τ_w from Equations 2.21 and 2.24 into Equation 2.20, we arrive at an approximate minimum value for t_{bit} of $12\mu s$. Accounting for the possibility that the actual value of τ_Δ might vary, we can define t_{bit} in terms of its relationship to τ_Δ:

$$t_{bit} = 10\mu s + \tau_\Delta / a, \tag{2.25}$$

where a is the ratio τ_Δ/τ_w.

Alternatively, Alice and Bob could choose not to synchronize their devices at all. In this case, Bob would simply always leave his detector on, and Alice and Bob would build their key by time tagging each photon transmission and detection. This method was not fully explored in the course of this work because time tagging each detected photon would be too memory-intensive for the microchips employed for the demonstration described in Chapter 3.

2.2.4 Secure Key Material Generation Rate B

Equations 2.19 and 2.20 can now be substituted into Equation 2.8 to evaluate the key material generation rate B as a function of the ratio $a = \tau_\Delta/\tau_w$,

$$\begin{aligned} B &= \frac{\lambda_{det}/2e^{\lambda_{det}}}{\tau_p + \tau_w} \\ &= \frac{\lambda_{src}P_{na}}{2}\left[\frac{\exp\left(-\frac{\lambda_{src}P_{na}\tau_\Delta}{a}\right)}{a\left(\frac{\tau_p}{\tau_\Delta}\right)+1}\right]. \end{aligned} \tag{2.26}$$

In addition, Equation 2.23 can be used to determine the relationship between I and a,

$$I_{max} = \frac{4\lambda_{src}U_{sys}\tau_\Delta}{a}. \tag{2.27}$$

The relationships of B and I to the ratio a are visualized together in Figure 2.3, which shows predicted results as τ_w increases (a decreases) for a QKD apparatus which generates single photons via Type-II SPDC (see Section 3.1.1). The system parameters displayed on the figure were intentionally chosen to be overly conservative. Even so, the results show that when τ_w is approximately an order of magnitude greater than τ_Δ, B reaches approximately 3kbps, which puts it just into the VLF spectrum range (3-30kHz). This rate is comparable to current NC2 systems. At this value, I remains between below 1%, meaning that any eavesdropper would only be able to gain that amount of the final key without being discovered (with the added bonus that at least 25% of the eavesdropper's version of the key would be garbled nonsense.)

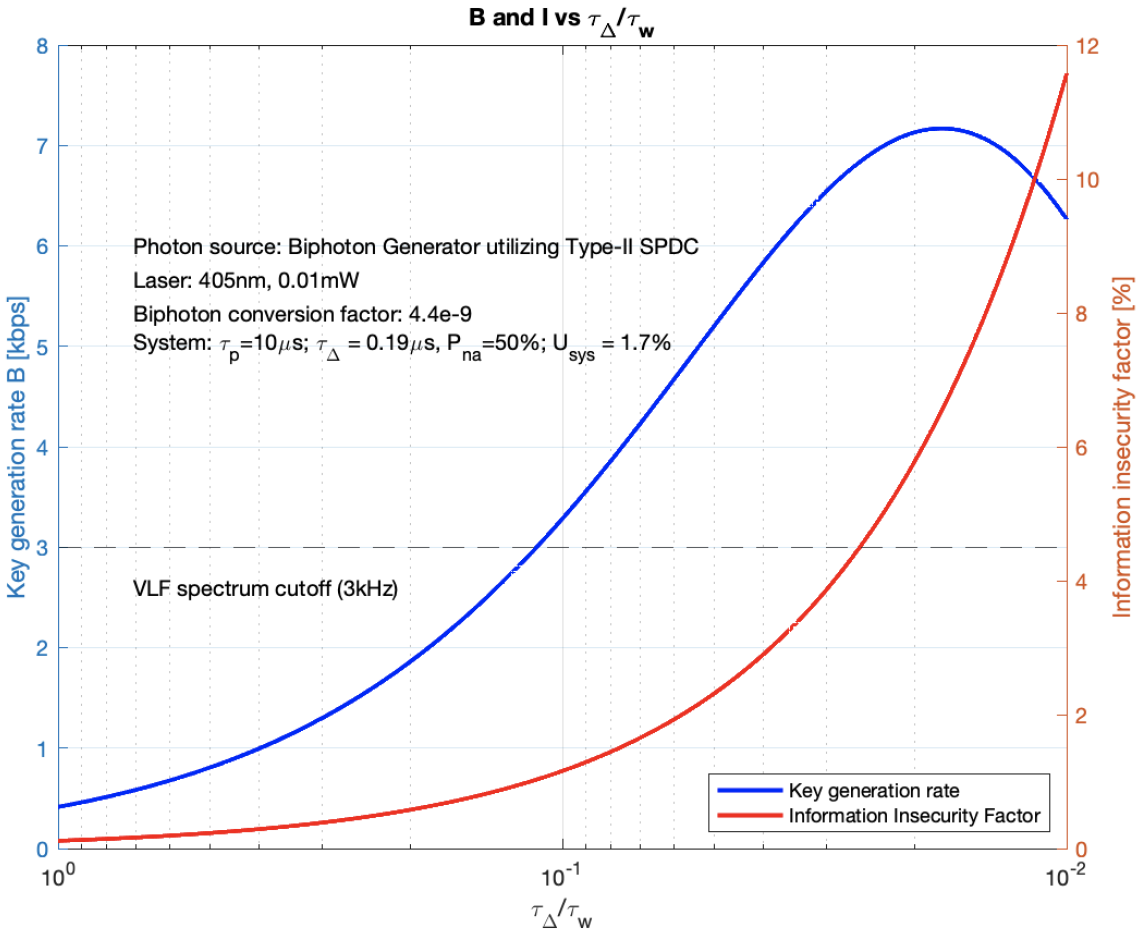

Figure 2.3. Relationship between secure key material generation rate B and information insecurity factor I to the ratio of the desync factor τ_Δ to the transmission window length τ_w for a hypothetical QKD apparatus based on Type-II SPDC. System parameters are chosen to be overly conservative. Results show that the apparatus can operate in the VLF range, which is comparable to current NC2 systems.

In short, Figure 2.3 shows that despite the difficulties associated with employing free space QKD, even conservative estimates indicate that QKD can realistically be employed in NC3 systems using currently available commercial/off-the-shelf technology.

CHAPTER 3:
QKD Demonstration Apparatus

The QKD demonstration apparatus was assembled in order to demonstrate the ease and affordability with which BB84 protocol can be carried out, and to lay the foundation for QKD research at NPS. The apparatus consists entirely of commercial, off the shelf optical and electronic components. This chapter presents a functional description of the demonstration apparatus, as well as calibration procedures and testing results.

3.1 Functional Description

The demonstration apparatus is divided into Station A, the sending station (or Alice,) and Station B, the receiving station (or Bob.)

A block diagram of the optical setup is shown in Figure 3.1. Pictures and screenshots of the QKD apparatus can be found in Appendix C. Tables 3.1 and 3.2 give functional overviews of each station.

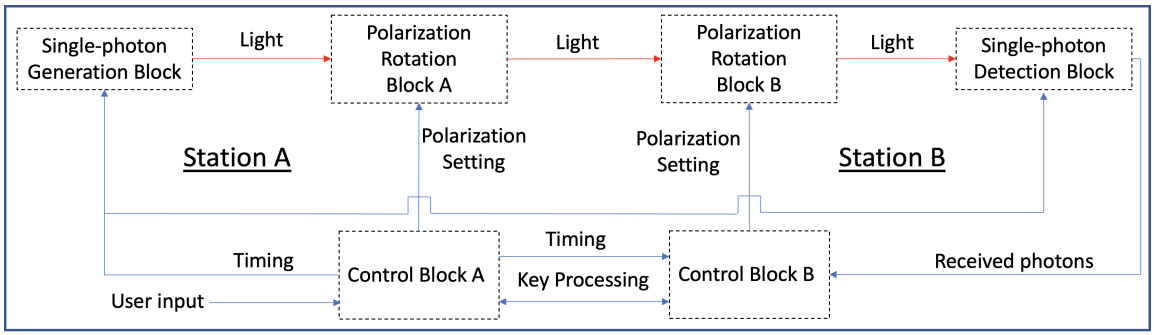

Figure 3.1. Block diagram of optical portion of QKD demonstration apparatus at NPS in Monterey, CA. Polarization Rotation Block B is not yet installed (see Section 4.1.)

Station A:

Functional Block Name	Function	Input	Output
Control Block A	Coordinate BB84 protocol	- Alice's Message - Bob's receiving bases and received states	- Raw key length (N) - Transmission state - Sending basis - Control pulses
Single-photon Generation Block	Generate "Single Photons on demand"	- Raw key length (N) - Control pulses	Polarized Single Photons
Polarization Rotation Block A	Rotate photon polarization states	- Polarized Single Photons - Transmission state - Sending basis	Alice's raw key bits

Table 3.1. Overview of Station A in the QKD Demonstration Apparatus. Includes both Key Generation and Key Processing functions.

Station B:

Functional Block Name	Function	Input	Output
Control Block B	- Choose receiving basis - Collect data from Station B - Communicate basis list and checking key to Station A	- Raw key length (N) - Measured states	- Measured state - Receiving basis
Polarization Rotation Block B	Place received photons in [R] or [D] polarization basis	- Polarized Single Photons - Receiving basis	Polarized Single Photons
Single-photon Detection Block	Measure received polarization states	- Polarized Single Photons - Control pulses	Measured polarization states

Table 3.2. Overview of Station B in the QKD Demonstration Apparatus. Includes both Key Generation and Key Processing functions.

3.1.1 Station A

Alice inputs the messages she would like to securely transmit to Bob at Station A. After receiving an input, Station A generates a raw key, which consists of randomly chosen 0's and 1's along with randomly chosen bases, of sufficient length to encrypt the message and to conduct post-transmission key processing and transmits that raw key to Station B. Station A then coordinates with Station B to process the raw key to ensure it was transmitted accurately and securely. After processing, Station A uses the final secure key to encrypt Alice's message and pass it to Station B.

Control Block A

Control Block A consists of Computer A, microcontroller A, and a pulse generator. A block diagram of Control Block A is shown in Figure 3.2.

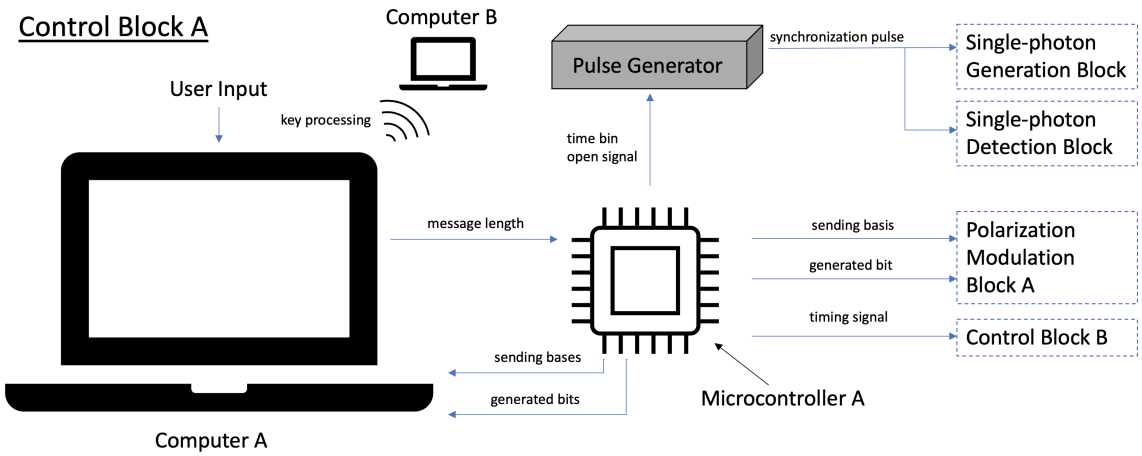

Figure 3.2. Control Block A. Consists of Computer A, Microcontroller A, and a pulse generator. Microcontroller A generally coordinates the process of generating a raw key. Once the raw key is generated, Computer A coordinates with Computer B to conduct key processing (see Section 3.1.3.)

Computer A. Computer A runs a Python script which reads Alice's message and then determines the length of the raw key N which must be generated to securely transmit that message to Station B. The script passes this length to microcontroller A through a USB port via Pyserial so that microcontroller A can begin the key generation process. Later, the script has Computer A coordinate with Computer B to conduct key processing .

Microcontroller A. Microcontroller A is an Arduino Nano BLE Sense microchip programmed in C++. Microcontroller A generally coordinates the key generation process.

Once the length of the raw key (N) has been determined, Computer A transmits this information to microcontroller A, which generates N secure bits. Bit generation proceeds as follows:

1. Microcontroller A generates a raw key bit (a random binary value, 0 or 1) and a random polarization basis (rectilinear, [R], or diagonal, [D].)
2. Microcontroller A passes the bit and chosen basis to Polarization Rotation Block A, which sets the polarization angle of transmitted photons.
3. Microcontroller A passes a Control Signal to the pulse generator.
4. The pulse generator passes a Control Pulse to the Single-photon Generation and Detection blocks, resulting in the transmission of one raw key bit from Station A to Station B.

This process is repeated until the number of bits generated is equal to N, the required length of the raw key. Once all of the required bits have been generated, microcontroller A passes Alice's raw key and the list of sending bases to Computer A for processing.

Pulse Generator. The Pulse Generator's function is simple: it generates a short Control Pulse whenever it receives a Control Signal. It passes the Control Pulse to an acousto-optic modulator (AOM) driver in the Single-photon Generation Block and the field-programmable gate array (FPGA) in the Single-photon Detection Block. While the pulse is active, the QKD apparatus is constantly transmitting and detecting photons; when the pulse ends, it stops. The Control Pulse is calibrated to be just long enough so that Station B detects exactly one photon approximately a third of the time (see Section 3.2.)

Single-photon Generation Block

The Single-photon Generation Block consists of a laser, a biphoton generator, a free space coupler, polarizing beam splitter (PBS) A, an AOM, a driver for the AOM, and a pulse generator. A block diagram of the Single-photon Generation Block is shown in Figure 3.3.

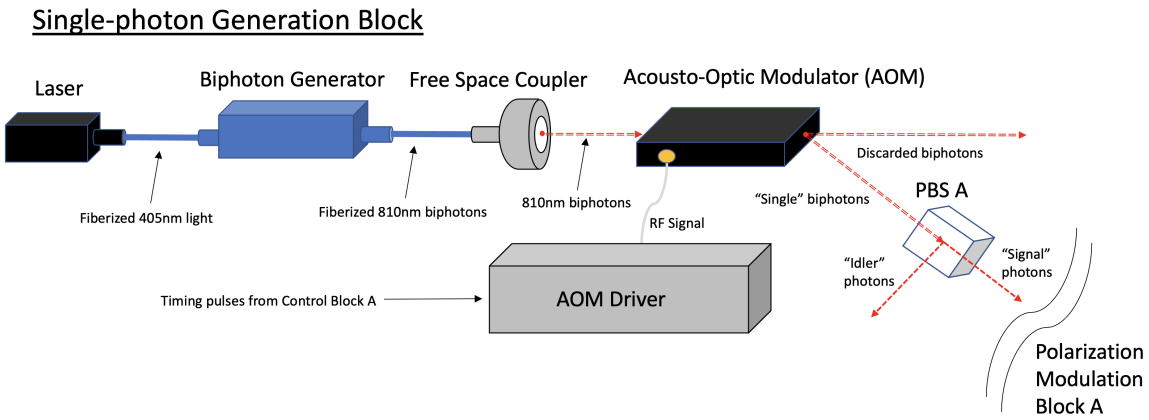

Figure 3.3. The Single-photon Generation Block. The laser and biphoton generator produce a steady stream of biphotons. When prompted by Control Block A, the AOM picks out one biphoton to send to the rest of the setup by deflecting the biphoton beam for a brief moment. The PBS picks one photon out of the pair and passes it to Polarization Rotation Block A for use in key generation.

Pump laser. The pump laser produces 7mW of coherent, polarized, 405nm light. The light is conducted via polarization-maintaining single mode fiber to the input port of a biphoton generator.

Biphoton Generator. The biphoton generator uses a periodically poled Potassium Titanyl Phosphate (PPKTP) crystal to convert 405nm light from the pump laser into 810nm light through a process called Degenerate Type II Spontaneous Parametric Down Conversion (SPDC).

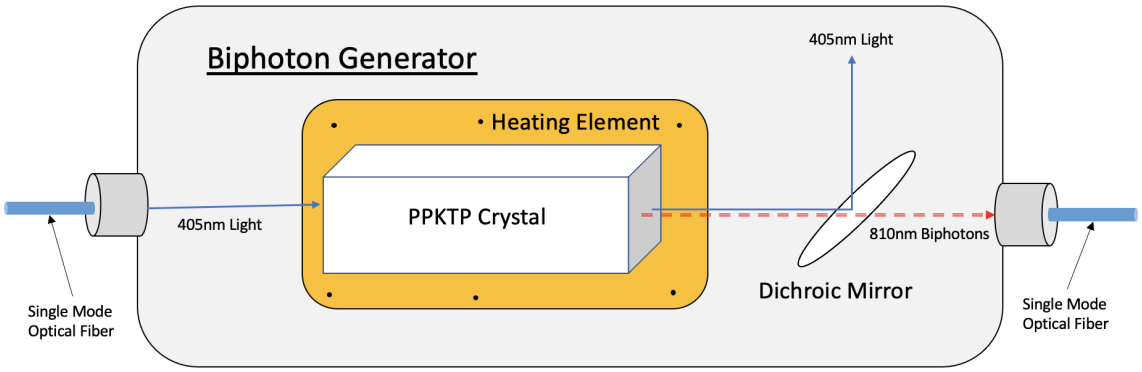

Figure 3.4. The biphoton generator. Adapted from *Quantum Mechanics Laboratory Kit Experiment Manual, Revision 2.0*, Qubitekk, Inc., December 2018. Incident 405nm light passes through a PPKTP crystal. Some of the light is absorbed and converted to 810nm biphotons via Type-II SPDC. The dichroic mirror blocks 405nm light and transmits 810nm light.

Degenerate Type II SPDC is a nonlinear process in which individual photons from a pump laser are converted into two photons historically labelled the "signal" and "idler" such that $\omega_p = \omega_s + \omega_i$, where ω_p is the frequency of the pump photon, ω_s is the frequency of the signal photon, and ω_i is the frequency of the idler photon. Since it is degenerate, $\omega_s = \omega_i$, so that $\omega_p = 2\omega_s$.[5] The signal and idler photons are collectively called biphotons, and they have the following interesting properties:

- The photons are created at nearly the exact same moment, colocated and in phase.
- Each photon has half the energy (and thus twice the wavelength) of the photon that was absorbed.
- The polarization state of the photons is a superposition of orthogonal states. When measured, one of them will always be found to have the same polarization angle as the photon that was originally absorbed, and the other will be found to have a polarization angle that is rotated by $\frac{\pi}{2}$ radians relative to the original photon.

The SPDC process has an efficiency of approximately 10^{-9}, so the 7mW of 405nm light is converted to 810nm biphotons with an output power on the order of hundreds of pW. A dichroic mirror at the output of the crystal reflects the 405nm light which is not converted and allows the 810nm biphotons to pass. The 810nm biphotons exit the biphoton generator through a single mode fiber. The fiber ends in a free space coupler which directs the biphotons to an AOM.

[5]Most biphoton pairs created by the biphoton generator are not perfectly degenerate. The degree of degeneracy of the biphotons is a function of the physical dimensions of the crystal, which are affected by the crystal temperature. In order to maximize degeneracy, the crystal in the biphoton generator is mounted on a heating element which maintains the crystal temperature within a narrow temperature range. Further discussion of the properties and degree of degeneracy of the biphotons can be found in Section 3.2.

AOM. The AOM functions as a fast shutter. It consists of a piezo-electric crystal whose index of refraction changes when a radio frequency (RF) signal is applied to the crystal. The AOM is oriented so that when no RF signal is applied, light from the free space coupler passes through it into a beam dump. When a radio frequency (RF) signal is applied, the light is refracted to a PBS. The RF signal is supplied by an AOM Driver and gated by Control Pulses from the Pulse Generator in Control Block A.

PBS A. PBS A is a polarizing beamsplitter cube which transmits horizontally-polarized light and reflects vertically-polarized light. When a degenerate biphoton pair encounters PBS A, one photon is transmitted and the other is discarded. The output of PBS A is a single polarized photon, which now exits the Single-photon Generation Block and enters Polarization Rotation Block A.

Polarization Rotation Block A

Polarization Rotation Block A consists of an EOM and High Voltage (HV) Circuit A. A block diagram of Polarization Rotation Block A is shown in Figure 3.5.

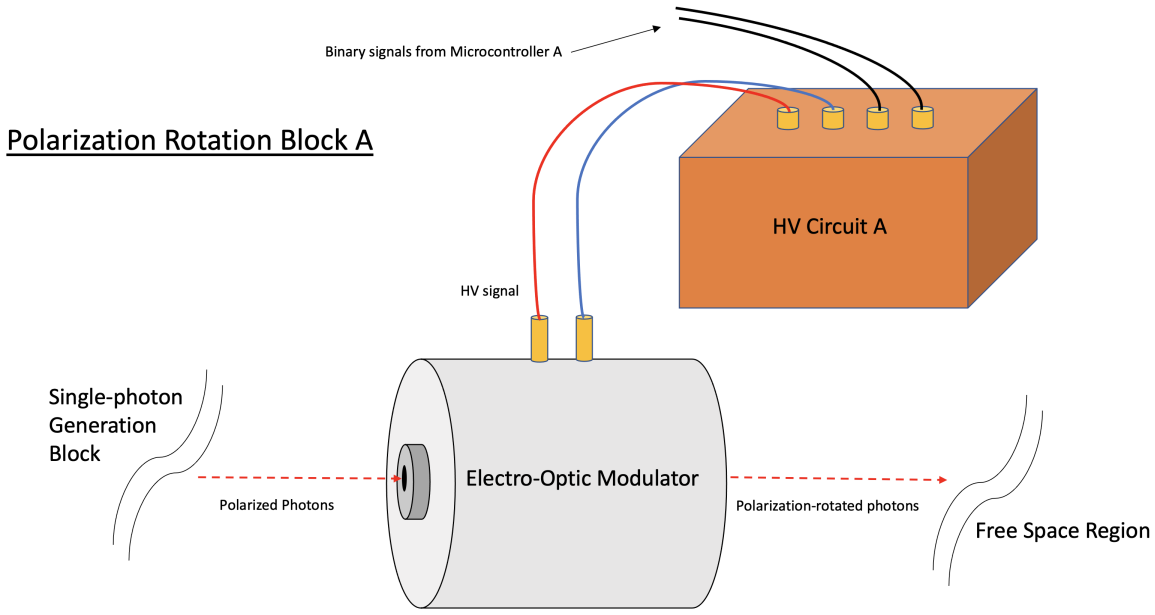

Figure 3.5. Polarization Rotation Block A. HV Circuit A sums the binary input signals from Control Block A and applies a DC voltage signal to the EOM. Polarized 810nm single photons from the Single Photon Generation Block pass through the EOM. The polarization angle of the incident photons is rotated through an angle proportional to the applied voltage.

EOM. The EOM is a transparent crystal which rotates the polarization angle of incident light by an amount proportional to the wavelength of the light and an applied DC voltage. Each EOM has a

characteristic half-wave voltage ($\frac{\lambda}{2}$ voltage) where it rotates the polarization angle of incident light of a given wavelength by exactly $\frac{\pi}{2}$ radians. The specific EOMs used in this demo are rubidium titanyl phosphate (RTP) crystals with a nominal $\frac{\lambda}{2}$ voltage of 715V for 633nm light.

HV Circuit A. HV Circuit A is a summing circuit which receives the two binary input signals from Microcontroller A and subsequently outputs one of four voltages: voltage setting 0, 1, 2, or 3. Each voltage setting is calibrated in relation to the EOM's $\frac{\lambda}{2}$ voltage. Voltage settings 0 and 2 correspond to 0V and the $\frac{\lambda}{2}$ voltage for 810nm light respectively. Voltage setting 0 allows incident light to pass without any effect, while voltage setting 2 produces orthogonally polarized light for an input signal which is aligned to the polarization axis of the EOM. Voltage setting 1 corresponds to one half of the EOM's $\frac{\lambda}{2}$ voltage, and voltage setting 3 corresponds to three halves of the EOM's $\frac{\lambda}{2}$ voltage. Voltage settings 1 and 3 produce orthogonally polarized light in the diagonal basis.

When light polarized along the axis of the EOM arrives from the Single-photon Generation Block, the EOM rotates the polarization of the light based on the last instruction it received from Microcontroller A. If the instruction was to send a 0 in the rectilinear basis, for example, voltage setting 0 is chosen, and the single photon from the Single-photon Generation Block is left in its initial polarization state. Voltage setting 1 corresponds to a 1 in the diagonal basis, and the polarization angle of the photon is rotated through an angle of $\frac{\pi}{4}$. Settings 2 and 3 correspond to a 1 and 0 in the rectilinear and diagonal bases, respectively, and the resulting polarization angles are $\frac{\pi}{2}$ and $\frac{3\pi}{4}$.

The output of Polarization Rotation Block A is a single polarized photon. The polarization angle of the photon relative to the axis of the EOM encodes one randomly generated bit in a randomly selected polarization basis. The photon exits and enters the free space region in between Station A and Station B.

3.1.2 Station B

Bob receives the transmitted raw key bits from Alice at Station B. Station B randomizes the basis of received bits, detects the polarization state of transmitted photons, and performs the logic filtering step of the key processing procedure.

Control Block B

Control Block B consists of Computer B and microcontroller B. A block diagram of Control Block B is shown in Figure 3.6.

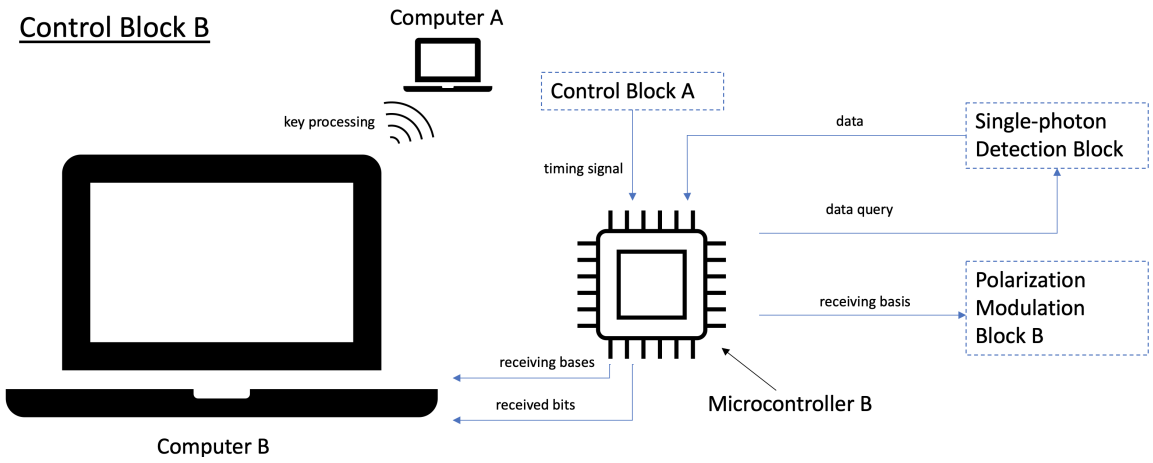

Figure 3.6. Control Block B. Microcontroller B takes timing signals from Control Block A and generally coordinates the actions of Station B during the raw key generation process. After the raw key generation process is complete, Computer B coordinates with Computer A to conduct key processing (see Section 3.1.3.)

Computer B. Computer B receives the length of the key to be generated from Computer A, then passes this length to microcontroller B. After key generation is complete, Computer B coordinates with Computer A to conduct key processing.

Microcontroller B. Microcontroller B is electrically connected to microcontroller A in Station A. Microcontroller A sends a signal along this connection when it is choosing a bit and sending basis. Each time microcontroller B detects this signal, it randomly chooses a receiving basis ([R] or [D]) and generates a corresponding binary signal which it passes to Polarization Rotation Block B. After the subsequent transmission event, microcontroller B signals the Single-photon Detection Block, prompting it to pass the number of detection events on each SPCM during that transmission event back to microcontroller B. Microcontroller B stores this information until the end of the key generation phase. Then, microcontroller B passes all the data to Computer B.

Polarization Rotation Block B

Polarization Rotation Block B sets the measurement basis for received photons. It consists of HV Circuit B and an EOM. A block diagram of Polarization Rotation Block B is shown in Figure 3.7.

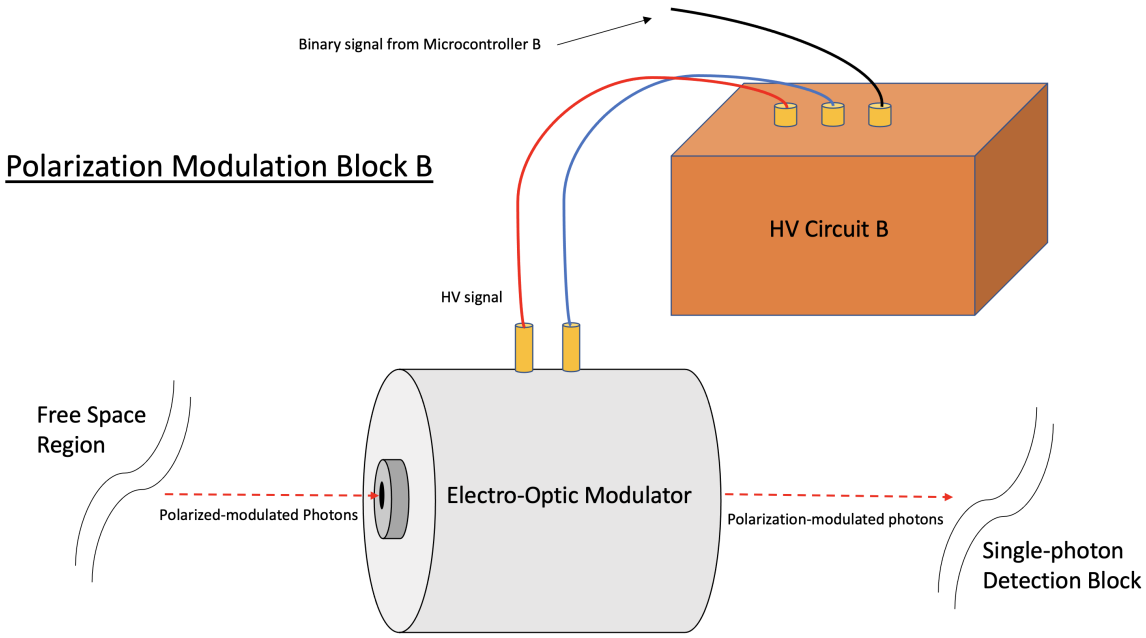

Figure 3.7. Polarization Rotation Block B. HV circuit B receives one binary signal from Control Block B. Otherwise, operation is the same as Polarization Rotation Block A (see Section 3.1.1).

HV circuit B. HV circuit B receives a binary input signal from microcontroller B and subsequently outputs one of two voltage settings, voltage setting a or b, respectively corresponding to 0V and half of the $\frac{\lambda}{2}$ voltage of the EOM. Voltage setting a corresponds to the rectilinear receiving basis, while voltage setting b corresponds to the diagonal receiving basis.

As discussed in Section 3.1.1, each photon arriving from Station A will have one of four possible polarization angles relative to the axis of the EOM in Polarization Rotation Block A: 0, $\frac{\pi}{4}$, $\frac{\pi}{2}$, or $\frac{3\pi}{4}$. When voltage setting a is applied to the EOM in Polarization Rotation Block B, photons pass through without their polarization angles being affected. When voltage setting b is applied, the EOM rotates the polarization angle of incident photons through an addition $\frac{\pi}{4}$ radians. Table 3.3 lists the possible input and output polarization states of photons from Station A passing through Polarization Rotation Block B.

HV Circuit A [Basis]	HV Circuit B [Basis]	Angle (rad)	Bit [Basis]
0 [R]	a [R]	0	0 [R]
1 [D]	a [R]	$\pi/4$	-
2 [R]	a [R]	$\pi/2$	1 [R]
3 [D]	a [R]	$3\pi/4$	-
0 [R]	b [D]	$\pi/4$	-
1 [D]	b [D]	$\pi/2$	1 [D]
2 [R]	b [D]	$3\pi/4$	-
3 [D]	b [D]	0	2π (0) [D]

Table 3.3. Tabulation of possible in/out states of the Polarization Modulation Blocks. [R] and [D] respectively refer to the rectilinear and diagonal bases. A "-" in the "Bit" column indicates that the sending and receiving bases do not match, and this bit will be discarded during key processing.

The output of Polarization Rotation Block B is the same as that of Polarization Rotation Block A: a single polarized photon, whose polarization angle relative to the axis of EOM A encodes a bit of information in a particular basis. Photons exiting Polarization Rotation Block B enter the Single-photon Detection Block.

Polarization Rotation Block B is not yet installed in the QKD demonstration apparatus due to a delayed equipment order.[6] Because this block is not yet installed, the apparatus cannot receive in the diagonal basis, and therefore it cannot fully implement BB84 protocol. It is currently configured to send and receive only in the rectilinear basis.

Single-photon Detection Block

The Single-photon Detection Block consists of a beam collimator, PBS B, SPCMs 0 and 1, and a FPGA. A block diagram of the Single-photon Detection Block is shown in Figure 3.8. Photons entering the Single-photon Detection Block are first collimated into a single mode optical fiber, which directs the photons to a fiberized PBS.

[6]The material similarity between Polarization Rotation Blocks A and B means that the exclusion of Polarization Rotation Block B does not significantly affect any of the quantitative analysis discussed in Section 3.2.

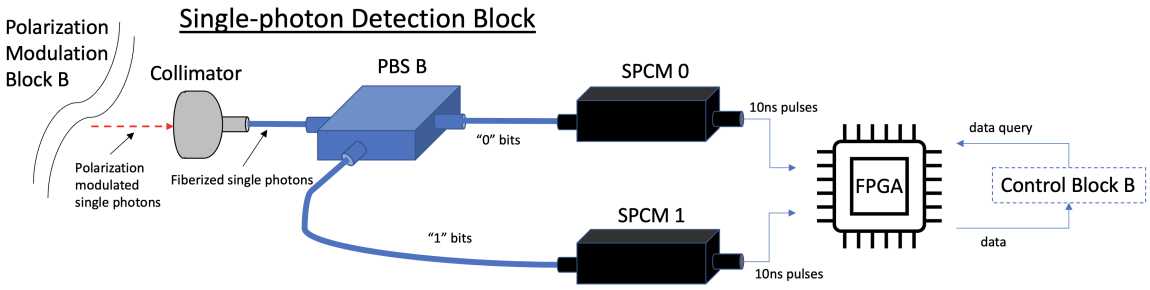

Figure 3.8. The Single Photon Detection Block. PBS B sorts incoming bits to SPCM 1 or 0 depending on their polarization. The FPGA detects pulses generated by the SPCMs as photons arrive, and sends this data along to Control Block B for processing.

PBS B. PBS B is a fiberized polarizing beamsplitter cube. PBS B is oriented so that incident photons with a polarization angle aligned with the axis of the pump laser (polarization angle $\theta = 0rad$) are transmitted to the input of SPCM 1. Orthogonally polarized photons ($\theta = \pi/2rad$) are reflected to SPCM 2. Photons of intermediate polarization angles are randomly distributed to either SPCM in accordance with the value of weighting factor $\cos^2(\theta)$.

SPCMs. The SPCMs are Silicon-based avalanche photodiode detectors (Si-APDs). When a photon in a particular wavelength range is incident on the semiconducting material of the detector, it is absorbed by an electron in the material's lattice. The excited electron leaves its spot in the lattice, creating a positively charged hole which attracts other nearby electrons, creating a small current. A high reverse bias voltage magnifies this current and causes the moving electrons to generate even more electrons via impact ionization. The resulting macroscopic current signals a photon detection.

FPGA. The electrical signals generated by the SPCMs travel via conducting wire to the input pins of a Field Programmable Gate Array (FPGA), which is an Intel microcontroller programmed in the HDL programming language Quartus II. It has a high clock speed, and thus is capable of detecting and recording the individual 10ns pulses coming from the SPCMs. When the FPGA receives a Control Pulse from the Pulse Generator in Control Block A, it begins recording the signals it receives from each SPCM. It continues recording for the duration of the pulse, then stops when the pulse ends. After the pulse ends, microcontroller B prompts the FPGA to send it the information it recorded, then clear its memory in preparation for the next bit.

3.1.3 Key Processing

As discussed in Section 2.1.1, key processing is the process by which Computers A and B convert the raw key generated by the optical setup into a secure key. Key processing starts when the length

of the raw key reaches N, the required length initially calculated by Computer A. At this point, the Python script which is running on Computer A prompts microcontroller A to pass Alice's raw key and list of sending bases to a file on Computer A. Microcontroller B also passes the SPCM detections and list of receiving bases to a file on Computer B.

Sifting step

In the sifting step (see Section 2.1.1, Computer A shares Station A's list of sending bases with Computer B, and Computer B shares Station B's list of receiving bases with Computer A. Because the bases contain no information about the transmitted bits or the detected photon states, this phase does not compromise any secure information.

Data usability. The first step in the sifting process is for Computer B to separate usable and unusable data. Usable data is any index corresponding to a transmission event where exactly one photon was received at either SPCM; unusable data is any index corresponding to a transmission event where no photons were received or the sum of photons detected on both SPCMs is greater than one. To determine data usability, Computer B scans the data file it received from microcontroller B. It records any usable single detection on SPCM 0 as a zero, and any usable single detection on SPCM 1 as a one. Computer B stores the resulting list of zeroes and ones as Bob's raw key. Computer B then shares the indices of unusable data with Computer A, and both computers discard the data associated with unusable indices.

Basis Matching. Table 3.3 shows that Station B can only receive useful information when the sending basis and receiving basis match. If they do not match, the photon's polarization angle will be intermediate when it reaches PBS B in the Single-photon Detection Block. To ensure that none of these basis-mismatched bits remain in the final key, Computer A and Computer B compare their lists of bases and discard all of the raw key bits corresponding to indices where the bases did not match. The remaining bits of each raw key, correlating to instances where the data received at Station B was usable and the sending basis and the receiving basis matched, form the sifted keys.

Checking step

In the checking step, Alice and Bob determine whether or not an eavesdropper may have attempted to intercept their communications. Computer A randomly chooses a set of bits from its sifted key and shares those bits with Computer B. It then removes those bits from its sifted key. The remaining bits form Alice's final key, and the removed bits form Alice's checking key. Computer B receives the list of indices from Computer A and also removes the bits associated with those indices from its sifted key, forming Bob's final key and checking key. The computers then compare the checking

keys against one another.

Error rate calculation. The checking keys are compared bit by bit. Each index where the checking key bits differ (for example, Alice's checking key has a 1 where Bob's checking key has a 0) is marked as an error. After the comparison is complete and the errors are tallied, the number of errors is compared to the length of the checked keys to calculate the error rate. As discussed in Section 2.2.2, this calculated error rate should be representative of the error rate of the still-hidden final keys.

Eavesdropper determination. As discussed in Sections 2.1.2 and 2.2.3, there is a characteristic error rate E for any real QKD system, measured prior to employment of the system for key generation, which will be used to alert the users to the presence of an eavesdropper. If the error rate measured during the checking step is higher than E, the generated key is not secure and is discarded. If the measured error rate is below E, the final keys are considered sufficiently secure for use in secure communications.

3.2 Calibrations

The QKD demonstration apparatus required various calibrations during setup. The results of those calibrations are described and documented here.

3.2.1 Circuit Timing

The first calibration was conducted to ensure that the control blocks were functioning as intended. An oscilloscope was used to determine to calibrate the timing and sequence of control pulses (see Figure 2.1 for the order of operation in a single bit transmission.)

Relative timing: control pulse and HV switching circuit

First, the output of the HV switching circuit's low voltage monitoring circuit (see Section 3.2.6) was attached to the oscilloscope in order to determine the switching time of the HV circuit. Then, Microcontroller A was programmed to wait for this time to elapse before sending the Control Signal to the pulse generator. Figure 3.9 shows the relative timing of the HV switching circuit (magenta) and Control Signal (blue).

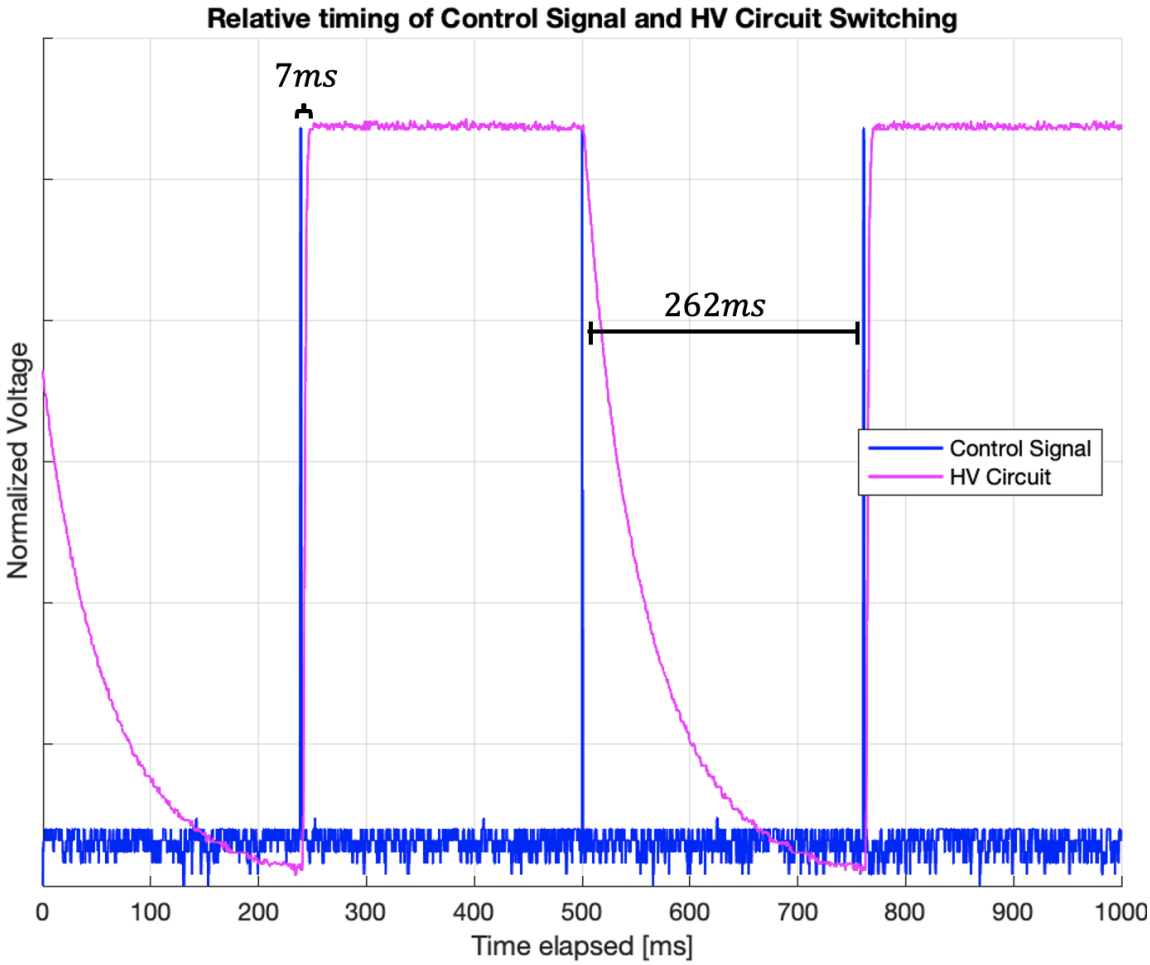

Figure 3.9. Oscilloscope output showing that the transmission window control pulses occur only after the HV circuit has stabilized.

Figure 3.9 shows that the HV switching circuit's rise time around is 7ms and its fall time is around a quarter of a second. These rise and fall times can be made several orders of magnitude smaller with commercially available equipment (see Table 4.1.) Because the voltage setting is chosen randomly for each transmission, Microcontroller A had to be programmed to wait for the longer of the two times. Microcontroller A was conservatively programmed to wait 270ms after sending the voltage setting to the HV switching circuit before generating the Control Signal.

Relative timing: detection window and transmission window

After the timing of the Control Signal was calibrated, the Control Signal was sent to the pulse generator, which was programmed to generate a single Control Pulse each time it received a Control Signal. The Control Pulse was sent to the Single-photon Detection Block to open the Detection Window and the Single-photon Generation Block to open the Transmission Window.

Both of these windows closed when the pulse ended.

- **Detection Window.** The Detection Window consisted of the FPGA receiving the Control Pulse and recording electrical signals from the SPCMs until the pulse ended. The FPGA's 50MHz clock speed was sufficiently fast that the time between generation of the Control Pulse and the opening of the Detection Window was negligible.
- **Transmission Window.** The Transmission Window consisted of the AOM Driver receiving the Control Pulse and generating a RF signal. As discussed in Section 3.1.1, the RF signal then traveled by wire from the AOM driver to the AOM and stimulated an acoustic wave. The acoustic wave spread outwards from the point of contact between the wire and the crystal, changing the crystal's index of refraction as it went. When the volume affected by the acoustic wave fully enclosed the region traversed by the biphotons from the Biphoton Generator, the AOM would deflect incident biphotons towards the rest of the apparatus.

The Detection Window is gated by the rising and falling edges of the Control Pulse. The Transmission Window is gated by the AOM diverting photons toward PBS A. Thus, the opening and closing of the Detection Window was observed by connecting the Control Pulse to an oscilloscope, and the opening and closing of the Transmission Window was observed by placing a photodiode at the output of the AOM and passing its DC signal to the oscilloscope as well. Figure 3.10 shows the lag time between the Detection Window (red) and Transmission Window (cyan) introduced by the relatively slow propagation of the acoustic wave through the AOM crystal.

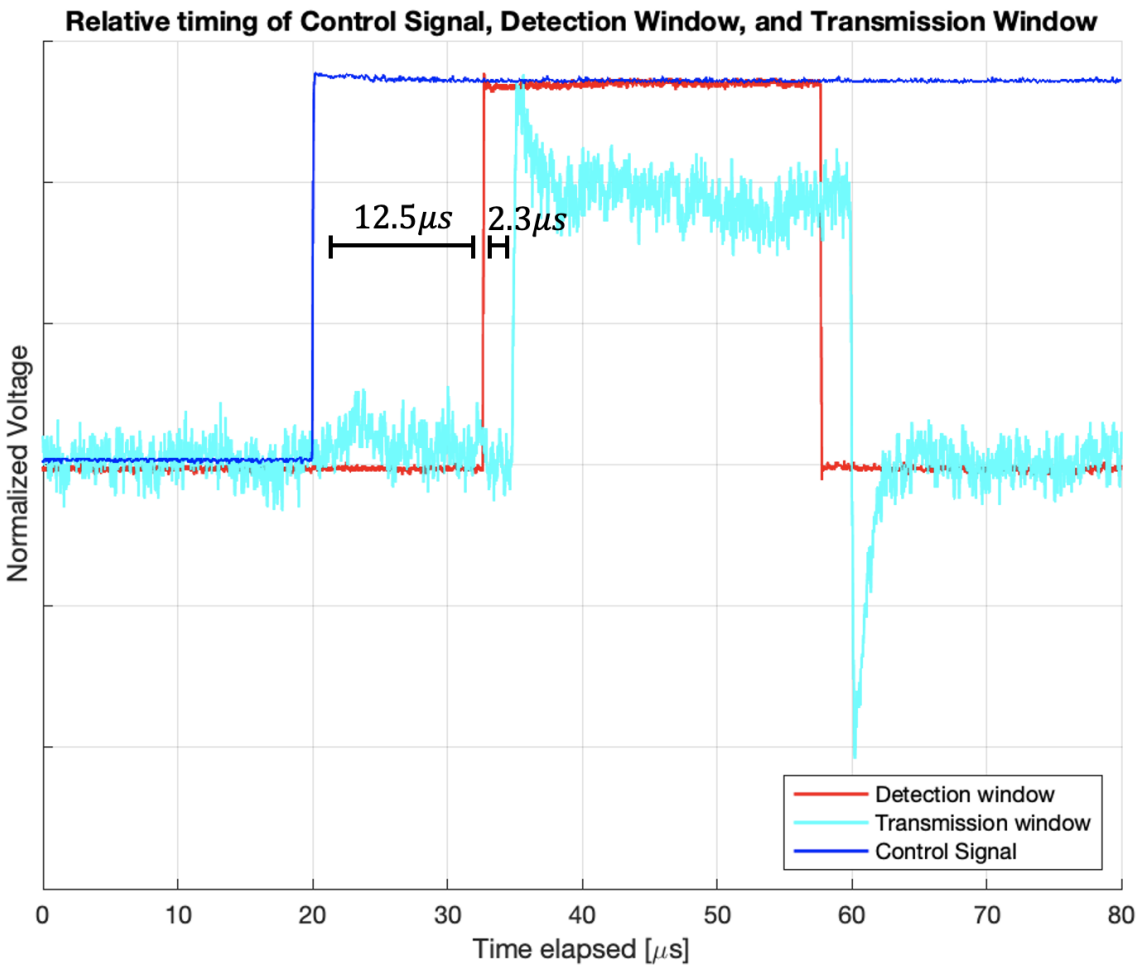

Figure 3.10. Oscilloscope output showing relative timing of the Control Pulse (blue), the Detection Window (red), and the Transmission Window (cyan)

The lag time was found to be 2.3µs, with no noticeable jitter. This value was used to set the value of the system's Desync Factor τ_Δ (see Section 2.2.3) and was subsequently used to determine the optimal Transmission Window length τ_w.

Note that the limits of wireless synchronization discussed in Section 2.2.3 do not apply to the apparatus because Station A and Station B are electrically connected. As such, the magnitude of τ_Δ could be significantly reduced by programming the FPGA to wait until a length of time equal to τ_Δ elapses after receiving a Control Pulse before opening the Detection Window (see Table 4.1.)

3.2.2 Synchronization

Of course, in an ideal free space QKD system, Stations A and B would be electronically separated and independent of one another. This would require efficient wireless clock synchronization between

Microcontrollers A and B in order to keep the transmission and detection windows synchronized. To this end, two Arduino Nano BLE microcontrollers were obtained and their clock speeds were compared under normal environmental conditions. Results are shown in Figure 3.11.

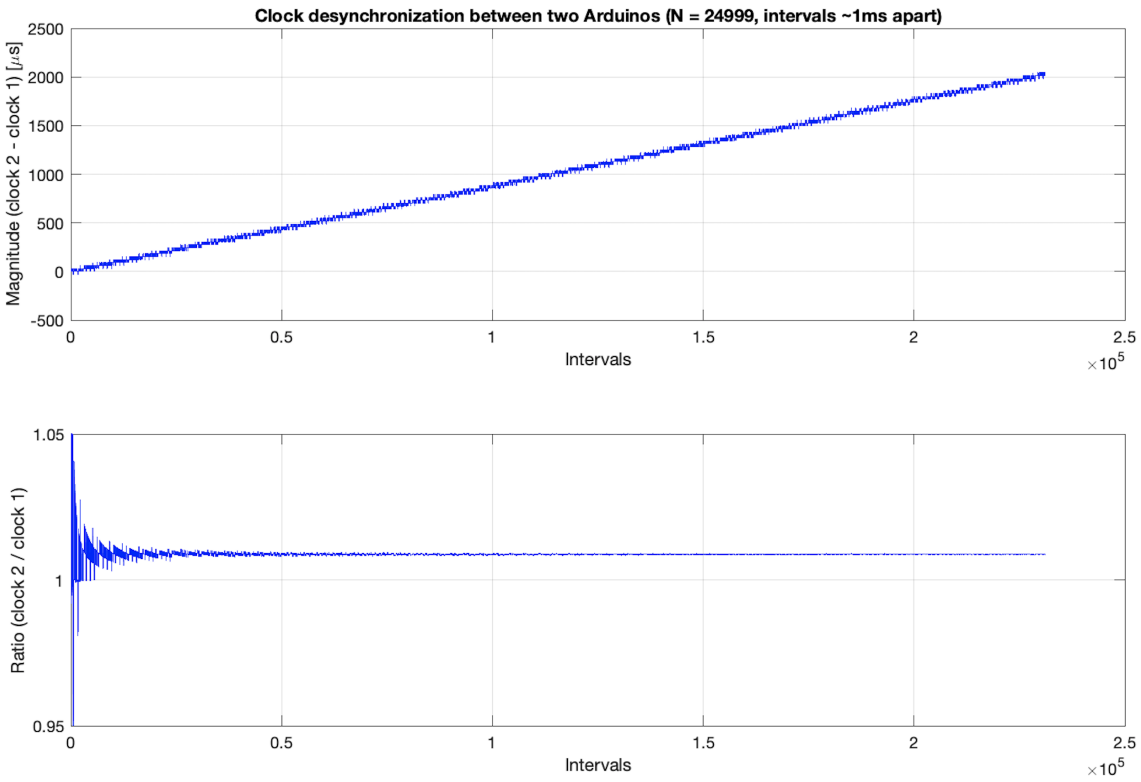

Figure 3.11. Comparing the clock speeds of two Arduino Nano BLE microcontrollers. Desynchronization between the clocks was found to be too rapid for use in the QKD demonstration apparatus.

The measured desynchronization speed between the two microcontrollers was approximately 1µs per 100µs, which would require the clocks to re-synchronize between each transmission in order to function properly. From this it was determined that the stations should be electronically connected and run from the same clock, and that electronic separation could be established later when microcontrollers with faster and more stable clocks could be obtained and programmed for the purpose (see Table 4.1.)

3.2.3 Counting Modules

The detectors used in the demonstration are internally cooled Silicon-based avalanche style photodiodes (Si-APDs), model number SPCM-AQRH-10-FC from the manufacturer Excelitas.

Due to certain inherent properties of Si-APD detectors, the number of electrical pulses generated

by the SPCMs may not reflect the number of photons transmitted by Alice with perfect accuracy. These properties are:

- **Detection inefficiency:** the SPCMs only detect approximately 60% of incident 810nm photons.
- **Reset time:** the electrical pulse generated by the avalanche has a 10ns duration, and the SPCMs require 12ns to reset between counting events, so an SPCM can only detect one photon arrival event every 22ns.
- **Background counts:** any light in the wavelength range of the SPCMs has a chance of being detected, including broadband photons generated from ambient lighting.
- **Dark counts:** thermal effects in the detector itself may cause electrons to jump energy levels, resulting in avalanche events which do not coincide with photon arrivals.

This section discusses how each of these effects is accounted for by the demonstration apparatus.

Missed counts

Detection Inefficiency. Figure 3.12 shows the efficiency curve from the specification sheet included with the SPCMs [24], showing the detector efficiency for 810nm light is approximately 60%. This results in a number of "empty" bins, where Alice may transmit a photon but Bob may not detect one. These empty bins are discarded during the sifting step, and thus they do not affect security, but they do negatively impact transmission efficiency.

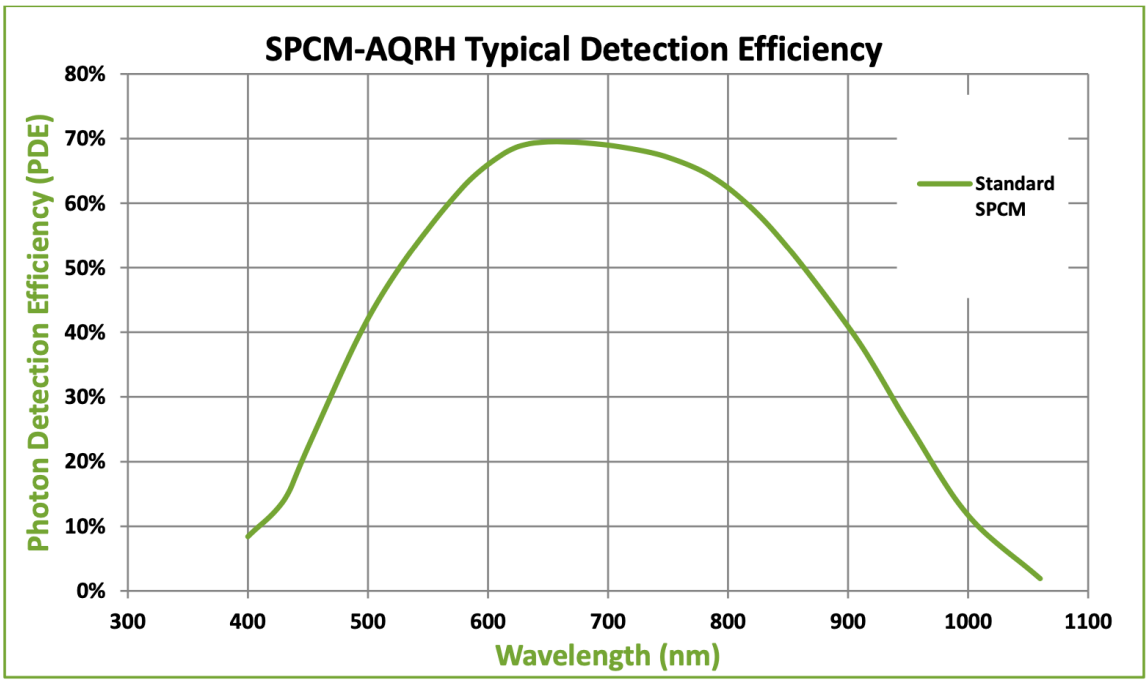

Figure 3.12. Detector efficiency curve for the SPCMs used in the demonstration apparatus.

Detector Reset Time. As will be discussed in Section 3.2.5, the photon source generates photons according to a random Poissonian distribution. The exact distribution was determined by connecting the fiberized Biphoton Generator output directly to a single SPCM. Microcontroller A was programmed to interface with the FPGA via the pulse generator and record pulses from each SPCM in time bins of length 500µs ($N = 10^6$.) Results are shown in Figure 3.13.

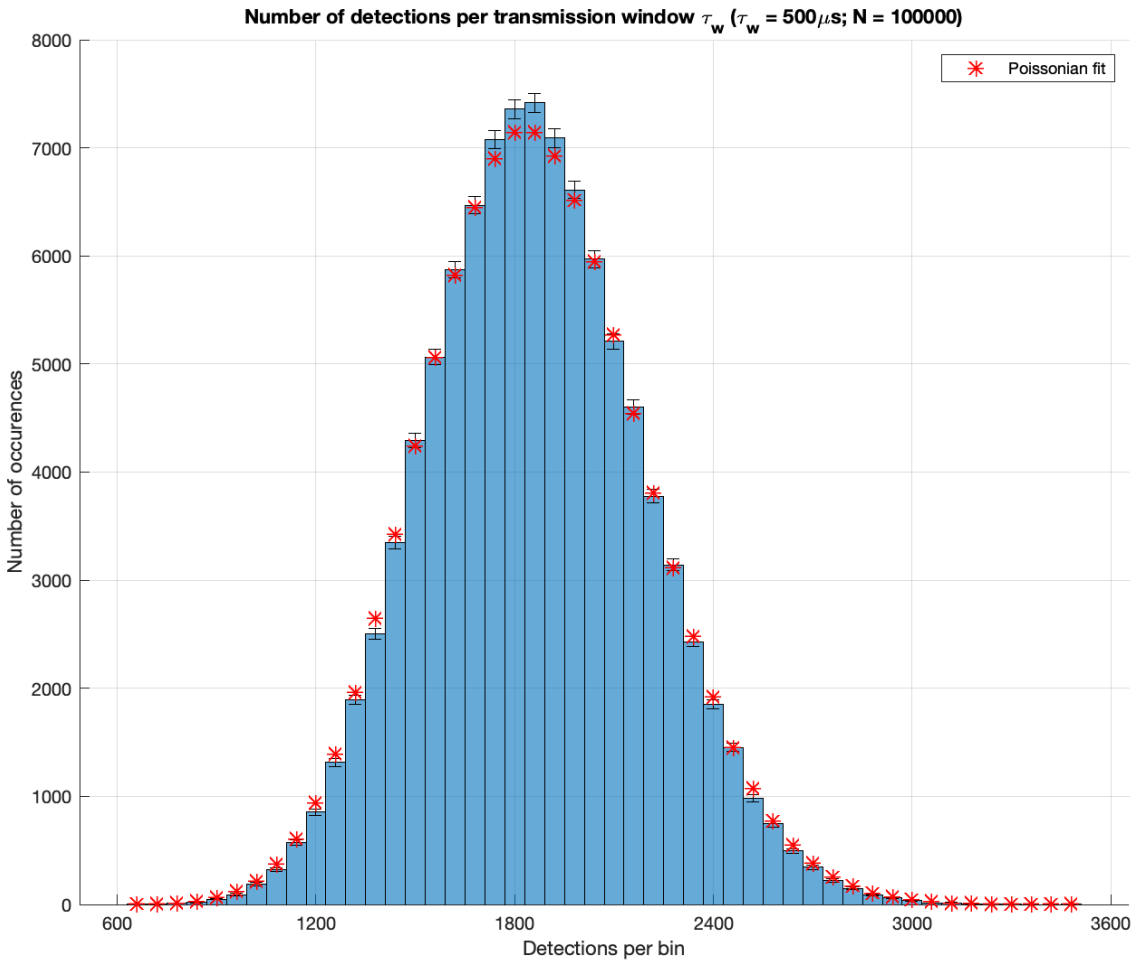

Figure 3.13. Histogram showing the number of counts per time bin of a detector counting incident photons from a Poissonian-distributed photon source transmitted via single mode fiber.

Figure 3.13 shows that the mean number of counts per second is $\lambda_{src} = 3.7 \times 10^6$, equating to a mean time of almost 270ns between each detection—an order of magnitude greater than the detector reset time of 22ns. Therefore, the effects of reset time on counting rate should be negligible.

Erroneous counts

Background Lighting. Background lighting was controlled by placing the detectors inside of a cardboard box with only a small hole cut out for the transmitted photons to enter (see photos in Appendix C.) The photon source was powered off and data was collected as per Section 3.2.3 once with the room lights on, once with the room lights dimmed, and once with the room lights off. Results are shown in Figures 3.14, 3.15, and 3.16 respectively.

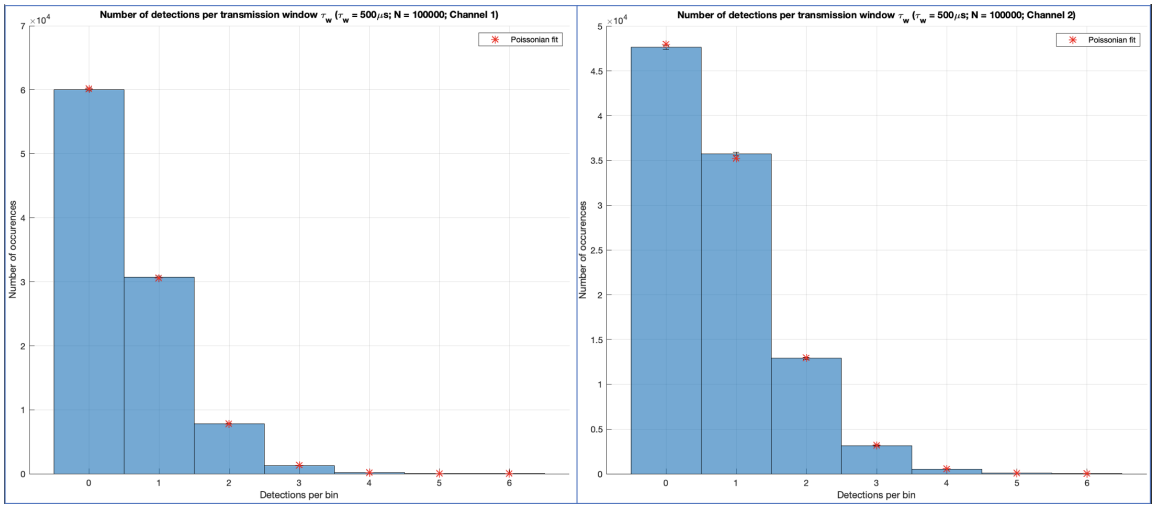

Figure 3.14. Histograms showing the number of counts per time bin of shrouded detectors counting dark counts and background counts in a bright room.

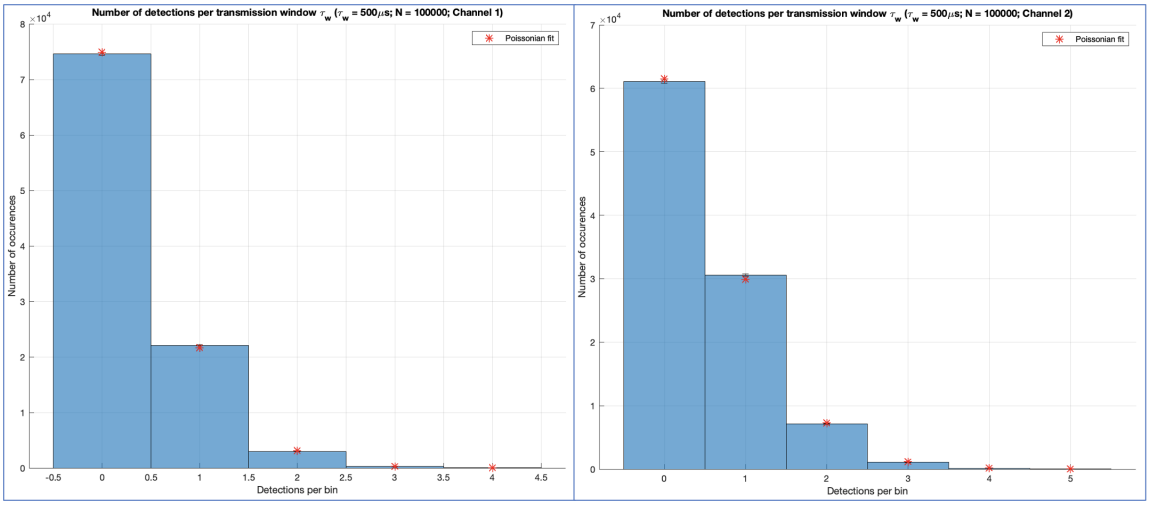

Figure 3.15. Histograms showing the number of counts per time bin of shrouded detectors counting dark counts and background counts in a dim room.

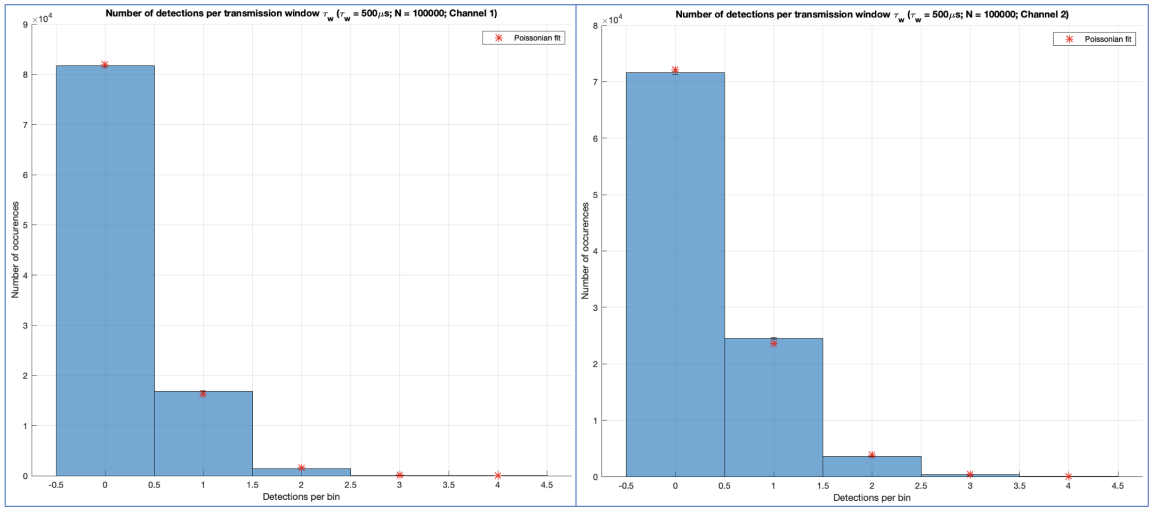

Figure 3.16. Histograms showing the number of counts per time bin of shrouded detectors counting dark counts and background counts in a dark room.

Comparison of the histograms resulting from each lighting condition show very little difference between "dark" and "dim" lighting conditions. Subsequently, background counts were minimized by always keeping the room dark or dim when conducting any operation with the demonstration apparatus.

Dark Counts. Nearly all of the counts displayed in Figures 3.15 and 3.16 are attributable to dark counts. When the photon source is powered on, the number of detections that are attributable to dark counts is very low compared to the number of detections that are attributable to the source (for example, compare Figures 3.15 and 3.16 to Figure 3.17, which shows the same measurements taken with the source powered on; the ratio $\lambda_{src}/\lambda_{dark}$ is very large.) Dark counts comprise only a small fraction of the total counts during operation of the QKD demonstration apparatus.

In summary, the four counting module effects are accounted for in the following ways:

- **Detector Efficiency.** Cannot be mitigated. Missed photons are accounted via sifting.
- **Reset time.** Counts missed due to detector reset time was determined to be a negligible portion of overall counts.
- **Background lighting.** Building a box around the SPCMs effectively blocked background lighting.
- **Dark counts.** Cannot be mitigated, but dark counts were determined to contribute only a small portion of the overall counts.

3.2.4 Optical Installation

The biphoton generator device used as the photon source in the demonstration was obtained by NPS from Qubitekk in October of 2019 as part of Qubitekk's Quantum Mechanics Lab Kit [28]. As shown in Figure 3.13, the source generates photons randomly following a Poissonian distribution. This distribution was checked at various stages while installing the optical portion of the QKD apparatus to ensure that the distribution of detected photons remained proportional to the distribution of photons generated at the source.

Free space, minimal equipment installed

After completion of the measurements described in Section 3.2.3, some components of the free space optical portion of the demonstration apparatus were installed. Specifically, the free space coupler and PBS A from the Single-photon Generation Block were installed, as well as the fiber coupler and PBS B from the Single-photon Detection Block. Data was then collected as per Section 3.2.3. Results are shown in Figure 3.17. Note that because there was no polarization switching (Polarization Rotation Block A had not been installed,) nearly all of the photons were detected by the same SPCM, in this case SPCM 1 (channel 2).

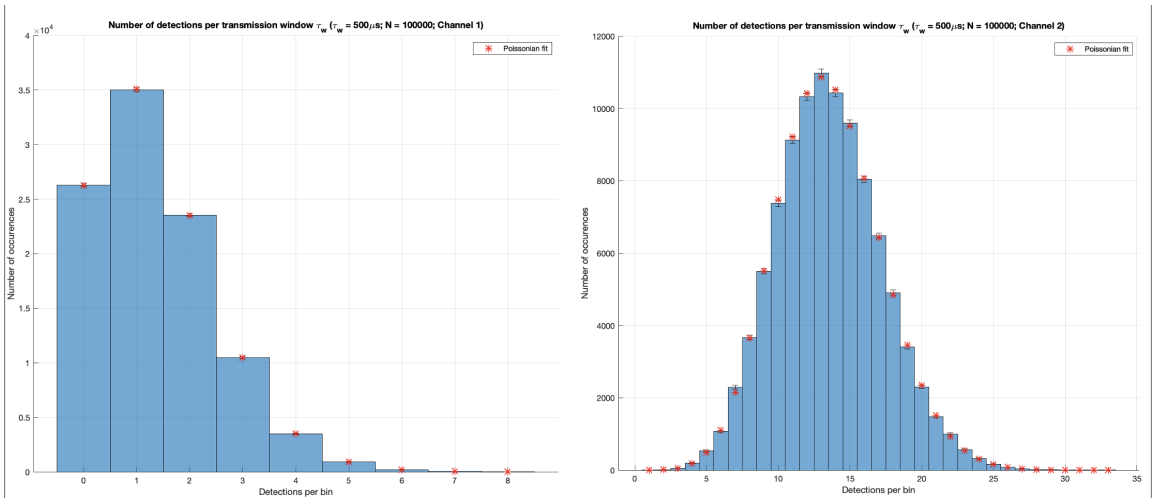

Figure 3.17. Histograms showing the number of counts per time bin of a shrouded detector counting incident photons from a Poissonian source transmitted over free space.

The distribution of detected photons is clearly Poissonian, but comparison of Figure 3.17 to Figure 3.13 shows the free space transmission incurs significant losses. The losses are primarily attributable to optical alignment and beam coupling/collimation: the photon source is too weak to be visible even with the assistance of sensitive IR cards and viewers, and so collimation was conducted with the help of a readily available 780nm laser. This method produced a beam collimation that was

sufficient for installing and testing the apparatus, but a collimation that is optimal for a 780nm beam is not necessarily optimal for a 810nm beam. This issue will be resolved upon the receipt and installation of a dedicated 810nm alignment laser (see Table 4.1.)

Free space, all equipment installed

After verifying that the detected photons which had been transmitted via free space followed the same statistical distribution as the source, the rest of the optical apparatus was installed and tested the same way. Specifically, the AOM and Polarization Rotation Block A were installed. Data was collected after installation, this time using a transmission window of 8.3µs and $N = 1.75 \times 10^6$, with Microcontroller A interfacing with both the FPGA and AOM via the pulse generator. Results are shown in Figure 3.18.

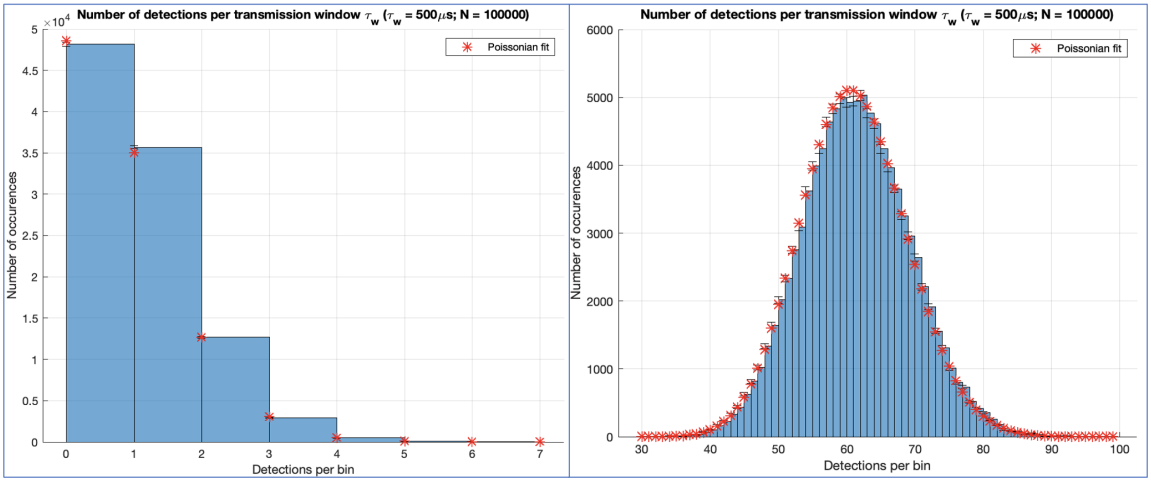

Figure 3.18. Histogram showing the number of counts per time bin of a detector receiving single photons from a Poissonian source through an optical apparatus set up for QKD.

Figure 3.18 shows that with all the equipment installed, detected photons still follow a similar statistical distribution to those found in Figures 3.13 and 3.17. Note that the mean value of the of the distribution in Figure 3.18 is greater than the mean value of the distribution in Figure 3.17. This increase was caused by some tweaking of the beam collimation of the free space couplers in between these two measurements, leading to an increase in the overall transmission efficiency of the free space portion of the apparatus.

In summary, the statistical distribution of photons transmitted through the QKD demonstration apparatus is linearly proportional to the Poissonian source distribution. Significant attenuation is introduced due to current limitations in available equipment and software.

3.2.5 Source Degeneracy

As discussed in Section 3.2.4, the Biphoton Generator generates a random Poissonian distribution of biphotons via degenerate Type-II SPDC. The resulting biphotons consist of one "signal" photon, whose polarization state is identical to the polarization state of the 405nm pump laser, and an "idler" photon, whose polarization state is orthogonal to the signal photon. The sum of the frequencies ω_s and ω_i of the signal and idler photons is equal to the frequency ω_p of the pump photons from the pump laser, such that $\omega_p = \omega_s + \omega_i$. For a perfectly degenerate SPDC process, $\omega_s = \omega_i = \omega_p/2$.

Degeneracy calibration

The Biphoton Generator's SPDC process is, in reality, only nominally degenerate. Actual values of ω_s and ω_i follow a random distribution, with a variance that is proportional to the SPDC process' degree of degeneracy, δ_D, where $\delta_D = \omega_s/\omega_i$.

δ_D is a function of the geometry of the PPKTP crystal, which in turn is affected by the temperature of the crystal, so that $\delta_D = \delta_D(T - T_0)$, where $\delta_D = 1$ (the SPDC process is perfectly degenerate) for some optimal crystal temperature T_0. As depicted in Figure 3.4, the crystal temperature is controlled by an internal heating element. Thus the degeneracy factor δ_D, and thus the variance in the frequencies ω_s and ω_i, is ultimately a function of the temperature setting of the Biphoton Generator's heating element.

As will be discussed in Section 3.2.6, the Polarization Rotation blocks operate best on light that is homogeneous and of a known wavelength; uncertainty in the wavelength of incident photons can result in errors. Therefore, it is important to ensure that δ_D is as close to unity as possible. In order to calibrate the Biphoton Generator for maximum degeneracy, the temperature of the heating element was varied and the resulting degeneracy factor δ_D was measured via a HOM experiment [29].

HOM experiment

The HOM experiment measures the degree of indistinguishability of photons, which in this case is identical to measuring the degeneracy factor. Figure 3.19 shows a diagram of the experimental setup. The HOM apparatus works by taking a biphoton pair from the output of the biphoton generator and sending the signal and idler photons down different optical paths, then recombining them at the input to a 2×2 50:50 beamsplitter at the end of the paths. The output of a 50/50 beamsplitter is random—a photon incident on either input port has an equal probability of exiting via either output port—but the HOM experiment shows that, if the beamsplitter cannot distinguish between two incident photons, both of the photons will follow the same random statistics. That is, indistinguishable photons incident on input ports 1 and 2 will both exit via the same output

port [29].

Indistinguishability factor δ_I. In order for two photons to be indistinguishable, they must be in exactly the same state. For the biphotons generated by the Biphoton Generator, there are three variables which must be controlled in order to ensure that the signal and idler photons are identical when they reach the 50/50 beamsplitter: they must have the same polarization state, they must be colocated, and they must have the same wavelength. These three factors are controlled as follows:

- **Polarization State**. Recall that Degenerate Type II SPDC creates two photons in orthogonal polarization states. In our HOM setup, the single mode optical fiber which transports the signal photons has a built-in $\pi/2$ rotation, leaving the signal photons in the same polarization state as the idler photons when they reach the beamsplitter.
- **Colocation.** The optical path which transports the idler photons has a built-in free space portion. The lens at one end is mounted on a motor which can move the lens back and forth along a 25mm track. The travel time of the idler photons can be tuned by adjusting the length of the free space portion. For some particular free space travel distance, the signal and idler photons can be made to arrive at the beamsplitter at exactly the same time.
- **Wavelength.** As discussed previously, the variance of the wavelengths of the signal and idler photons is a function of the degree of degeneracy of the SPDC process, which in turn is controlled by the temperature setting of the heating element in the Biphoton Generator. The ultimate goal of this calibration was to find the temperature setting which would minimize the variance in the biphoton wavelengths.

The HOM apparatus shown in figure 3.19 shows how these factors were accounted for in order to measure the indistinguishability factor δ_I.

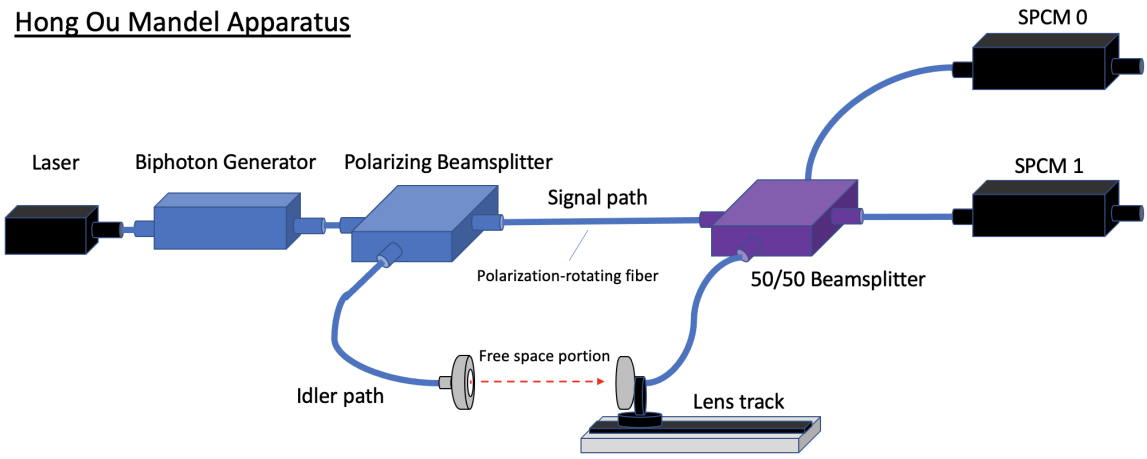

Figure 3.19. HOM apparatus built to calibrate the Biphoton Generator's internal heating element temperature setting for maximum biphoton degeneracy.

Method. We set up the HOM apparatus in accordance with Figure 3.19. We initially left the Biphoton Generator's internal heating element at its manufacturer-recommended default setting of 45.42°C, then adjusted the free space portion of the idler photon path length so that indistinguishability factor δ_I would be very low.

Preliminary data. We collected preliminary data from SPCM 0 and 1 as follows: SPCM output pulses were observed ten times in increments of 100ms each. A "coincidence count" was recorded whenever both SPCMs recorded a detection event within the same brief "coincidence window" (3ns.)

Coincidence counts correspond to events where signal and idler photons from the *same* biphoton pair arrive at the input of the 50/50 beamsplitter and subsequently exit via *separate* output ports. This happens when the photons in the biphoton pair are distinguishable. For the preliminary data run, the photons were distinguishable because their travel times were different, meaning that they were not colocated when they arrived at the input to the beamsplitter. Therefore, they each followed their own random statistics: half the time, they each went to the same output (and no coincidence count was recorded), and half the time they each went to different outputs (resulting in a coincidence count.) Note that the coincidence count rate is maximized when the biphotons are very distinguishable, and minimized when the biphotons are indistinguishable.

The preliminary data was used to set the baseline coincidence count rate C_0. Once the baseline rate was established, we began moving the motor-mounted lens in increments of 0.05mm. For each increment, we recorded coincidence counts for 100ms and calculated the coincidence count rate C.

The indistinguishability factor δ_I for each position was calculated as

$$\delta_I = \frac{C_0 - C}{C_0} \tag{3.1}$$

and plotted against the position of the motor on its track. Results are shown in Figure 3.20.

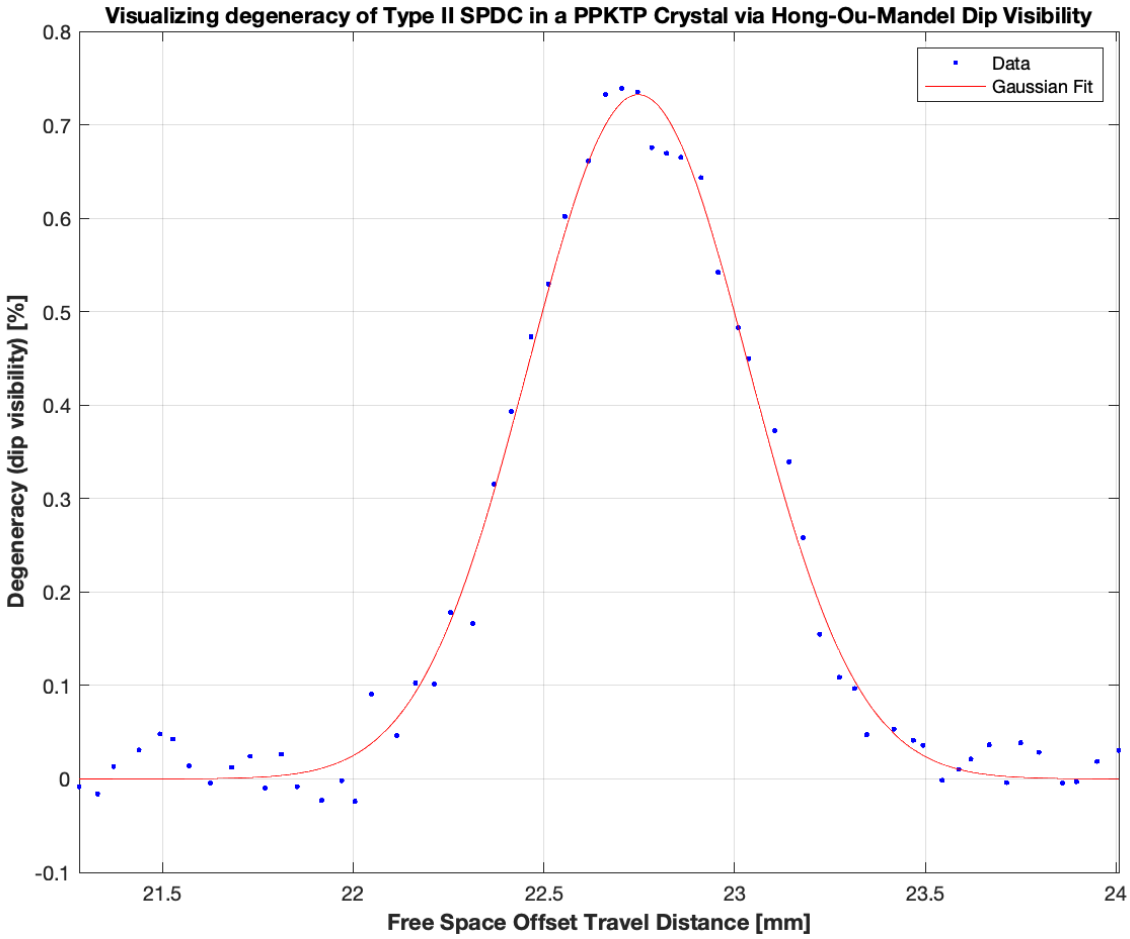

Figure 3.20. HOM experiment results showing the optimal path length setting for source degeneracy using manufacturer's recommended settings.

Figure 3.20 shows that, using the manufacturer's default temperature setting, the maximum achievable value of δ_I was around 0.67.

Optimal temperature determination. In order to find the optimal temperature setting for maximum degeneracy, we ran the HOM experiment many times, varying the heating element temperature setting between each run. Figure 3.21 shows the maximum value of δ_I recorded for each temperature

setting.

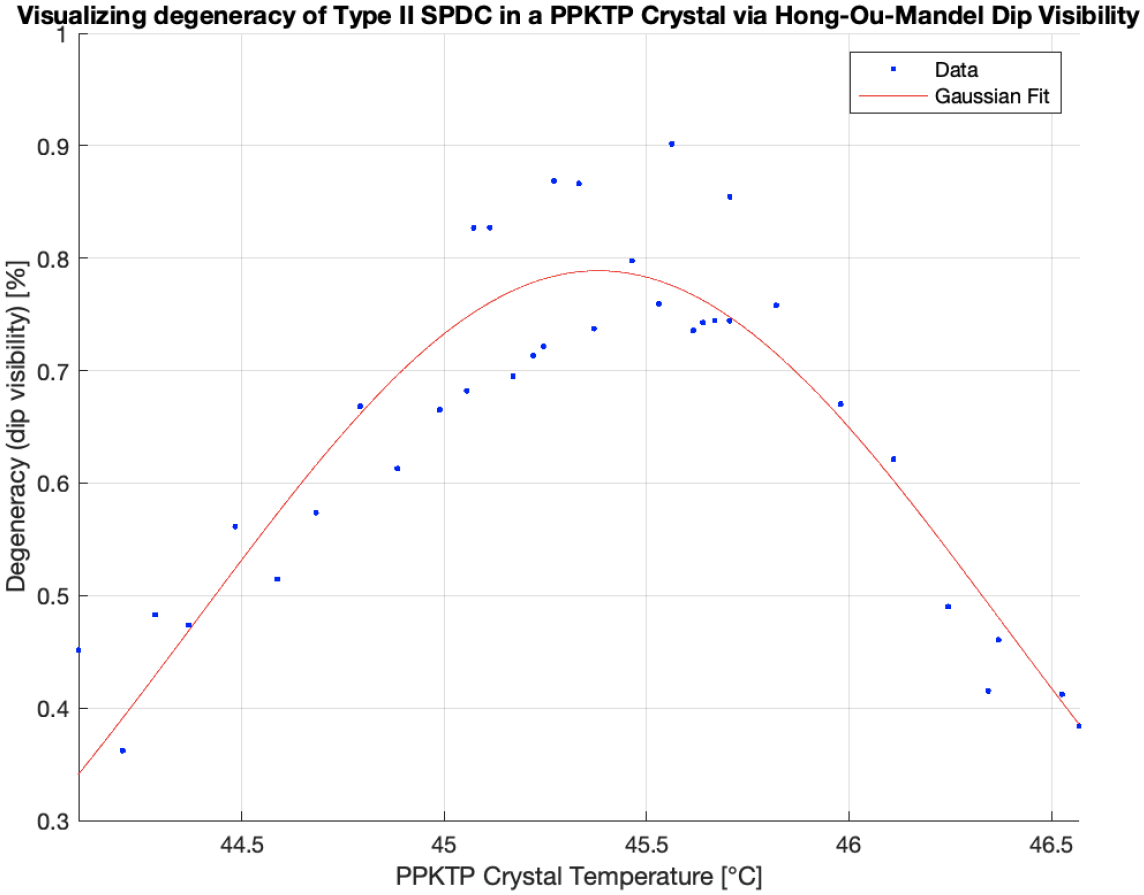

Figure 3.21. HOM experiment results showing the optimal PPKTP crystal temperature setting for source degeneracy.

The results of Figure 3.21 were used to determine the optimal internal heating element temperature setting corresponding to maximum degeneracy factor δ_D of the SPDC process in the Biphoton Generator. Note that even when the position and temperature were both optimized for maximum indistinguishability, δ_I did not approach unity. We believe this could be attributable to inefficiencies in the optical alignment and fiber couplings, inaccuracies in the polarization-rotating fiber used to match the polarization states of the biphotons, or the bandwidth of generated photons.

3.2.6 HV Circuit Settings

HV Circuit A consists of a tunable voltage source, a low voltage (LV) monitoring circuit, four rheostats (each corresponding to a specific voltage setting—see Section 3.1.1), and a logic circuit. The logic circuit receives input from Microcontroller A in the form of two binary signals, corresponding to Alice's choice of bit value and basis for each photon transmission event. Based on the

input it receives, it selects the rheostat corresponding to voltage setting 0, 1, 2, or 3, each of which is tuned to different setting for the voltage source. Once the selection has been made, the voltage source applies the chosen voltage to the attached EOM, which rotates the polarization angle of incident light proportionally to the applied voltage.[7]

The EOM model used in the apparatus is a Rubidium Titanyl Phosphate (RTP) pockels cell from the now-defunct New Jersey based laser equipment supplier Fastpulse Technology. In order to determine the EOM's calibration curve for 810nm light, we selected one voltage setting rheostat from HV Circuit A and tuned it to vary the voltage applied to EOM A. While varying the voltage, we collected data from SPCMs 0 and 1 (channels 1 and 2) in the manner described in Section 3.2.3 using a transmission window of 800μs (N = 5000). Figure 3.22 plots the results of this measurement for each SPCM, as well as the ratio of the two average counts on each SPCM, against the output of the LV monitoring circuit.

[7]That is, the output signal of the EOM has a polarization state which is a superposition the polarization state of the incident light and the associated orthogonal states (circular polarization included) with weighting factors for each state varying as a function of the magnitude of applied voltage. For these measurements, the only voltage settings of interest were those corresponding to states where the weighting factor of the circular polarization state was zero, such that the overall polarization state was a superposition only of rectilinear states.

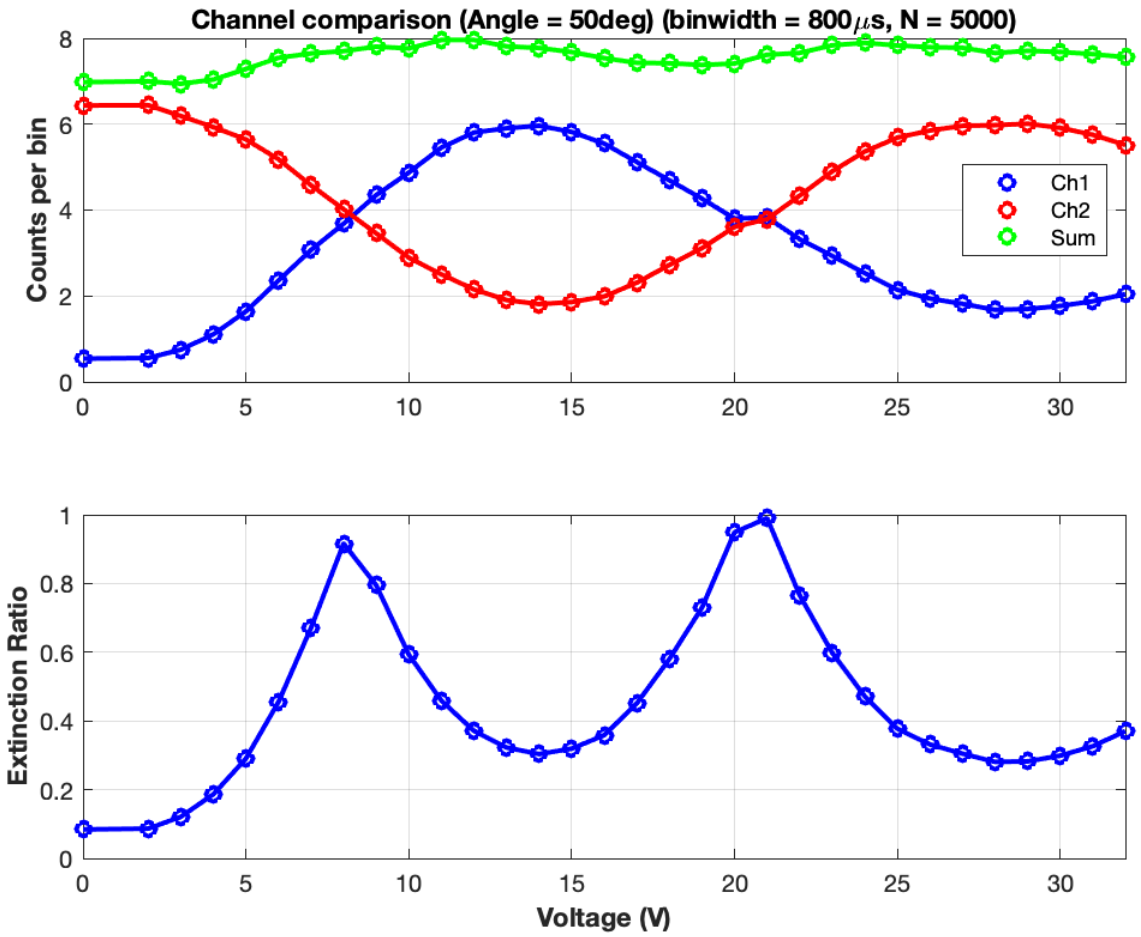

Figure 3.22. Results of the calibration measurements described in Section 3.2.6. Top: average counts per time bin for channels 1 and 2 (SPCMs 0 and 1) plotted against the output of a low voltage (LV) monitoring circuit attached to HV Circuit A. Bottom: the ratio of the lower of the average counts per time bin from SPCMs 0 and 1 to the higher. The local maxima and minima on this plot give the LV circuit output corresponding to the optimal HV Circuit A rheostat position for voltage settings 0, 1, 2, and 3 (see Section 3.1.1.)

The data collected via these measurements reveal the optimal rheostat positions for HV Circuit A corresponding to voltage settings 0, 1, 2, and 3: the minimums on the plot correspond to voltage settings 0 and 2 (the rectilinear sending basis), while the maximums correspond to voltage settings 1 and 3 (the diagonal sending basis).[8] Similar calibrations are planned for HV Circuit B when a second voltage source is obtained (see Table 4.1.)

[8]The exact HV circuit voltage settings were not measured precisely, due to the convenience and utility of simply monitoring the LV circuit output. However, the HV settings are known from manufacturer specifications to range up to 1.2kV for the observed polarization angle rotations.

3.3 Results

After all of the calibrations and installations described in Section 3.2 were complete, the apparatus was tested to determine its functionality and system error rate.

3.3.1 Sample Output of Demonstration Apparatus

To test the apparatus, "Alice" entered a short message into Computer A, then used the demonstration apparatus to conduct QKD and distribute the message to "Bob." After the key generation and key processing phases were complete, Alice's version of the generated key was used to encrypt the message, then Bob's version of the key was used to decrypt the message.[9] The total time, transmission efficiency, and true error rate of the transmission were also calculated. The results of one trial are shown in Figure 3.23.

[9]The encryption algorithm implemented was fairly rudimentary: for each character of the message, three successive digits of the generated key were treated as a binary number and added to the ASCII code of the character. For decryption, Bob did the same process in reverse.

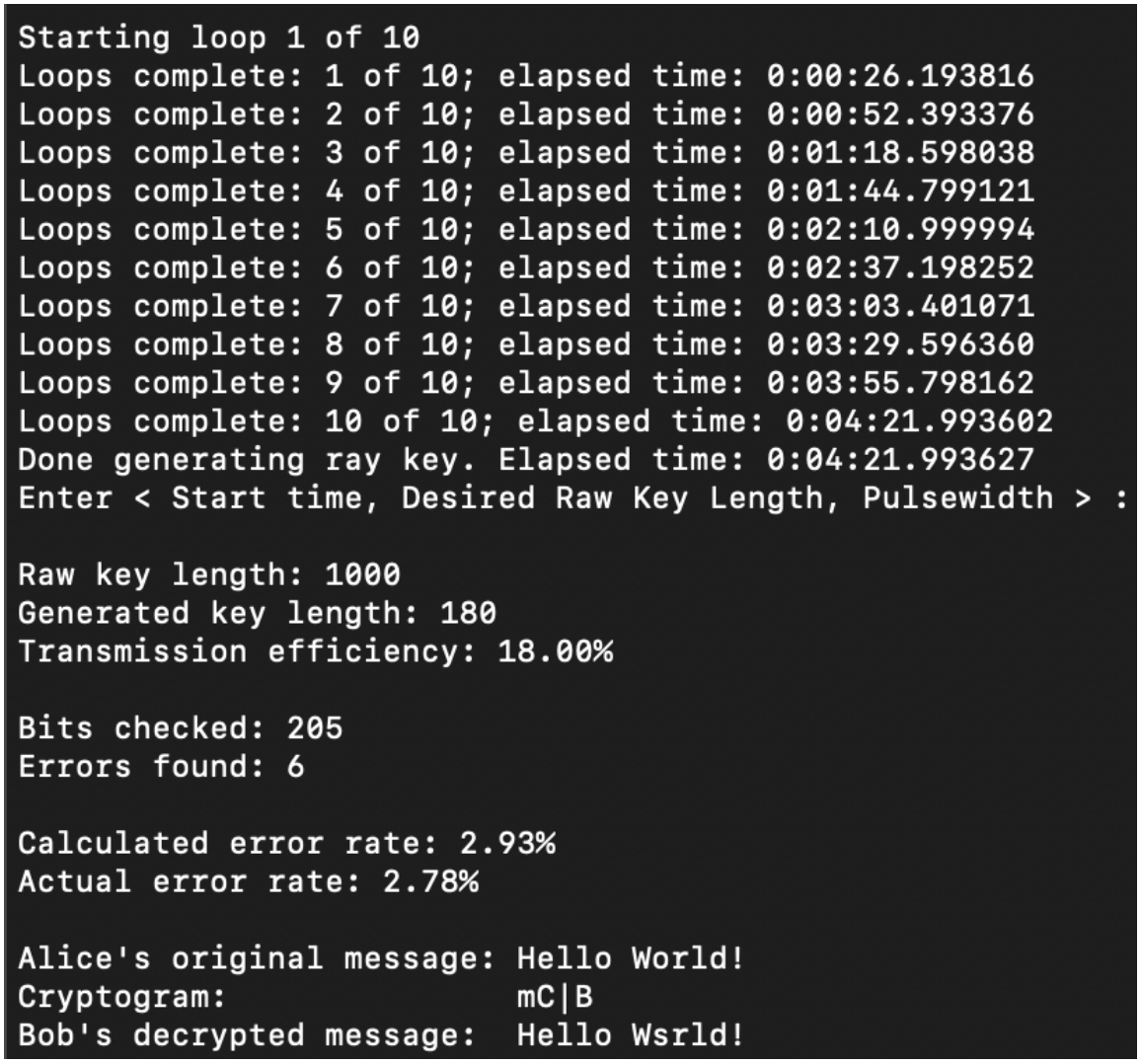

Figure 3.23. Sample output of the QKD demonstration apparatus described in Section 3.1.

Figure 3.23 shows that in order to send the message “Hello World!”, Alice generated 1000 raw key bits. The final key was 180 characters long and took 4 minutes and 21 seconds to transmit. The checking phase determined that the error rate was 2.93%, which was very close to the actual error rate, and well below the 25% error rate threshold which might indicate a clumsy intercept-and-replace attack from Eve. The final message as deciphered by Bob differed from Alice’s original message by one character, but was still readable. The results of this test indicate that the apparatus is capable of distributing information by encoding the information in quantum states, and that the information it distributes is usable.

3.3.2 Apparatus Statistics

Subsequently, the system error rate and transmission efficiency were measured along with their associated uncertainties. The measurement consisted of running the apparatus continuously for several hours. In all, 350 different keys were generated throughout the test, each consisting of 500 raw key bits. Figure 3.24 displays the error rate and transmission efficiency results.

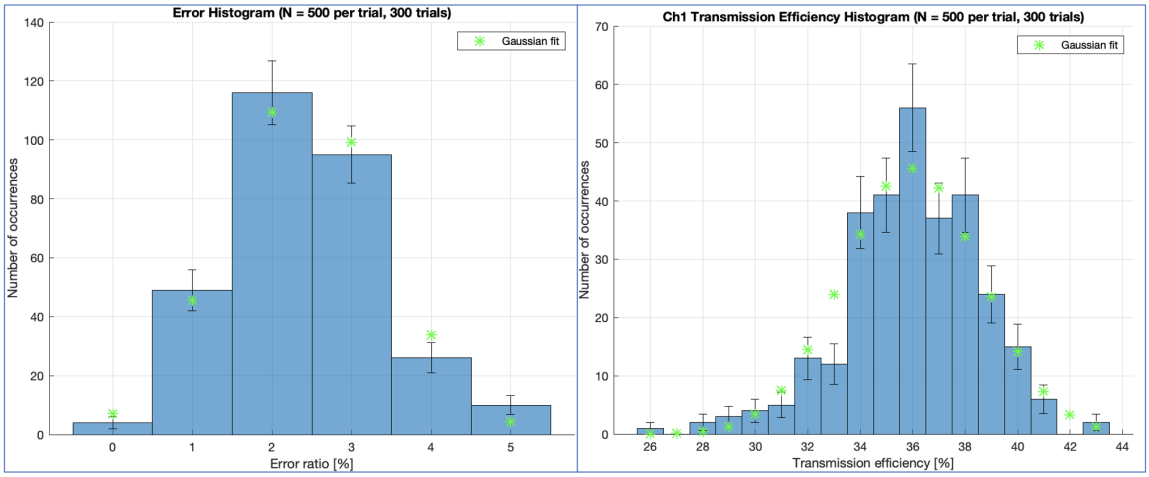

Figure 3.24. Experimental results of a long-duration error rate analysis of the QKD demonstration apparatus described in Section 3.1. Left: error rate plotted for each trial. Right: transmission efficiency plotted for each trial.

From the data collected, the system error rate and transmission efficiency were calculated to be $2.36\% \pm 1.04$ and $35.98\% \pm 2.63$ respectively. Substituting $U_{sys} = 0.0104$ into Equation 2.27, along with the values $\tau_\omega = 8.3\mu s$ and $\lambda_{src} = 3.7 \times 10^6$ from Sections 3.2.1 and 3.2.5, we calculated the Information Insecurity factor I for the apparatus:

$$I_{max} = 4\lambda_{src}\tau_\omega U_{sys} = 1.3 \tag{3.2}$$

The fact that I is greater than unity signifies that Eve may execute intercept-and-replace attacks against any amount of the transmitted key and remain undetected. This is mostly because Alice currently transmits over 30 photons per transmission event ($3.7\times10^6 \frac{\text{photons}}{\text{sec}} \times 8.3\times10^{-6} \frac{\text{sec}}{\text{transmission}} = 30.7 \frac{\text{photons}}{\text{transmission}}$), thus giving Eve ample opportunity to intercept and replace one without being detected.

Transmission efficiency

Alice must transmit so many photons because of the poor free space coupling of the apparatus. Consider the averages of the count distributions shown in Figure 3.13 ($\lambda \approx 1830$) and Figure 3.18 ($\lambda \approx 61$). The only major difference between these two measurements was that the free space portion was added in between. The optical attenuation factor of the free space portion is thus $61/1830 = 0.034$, and the average number of transmitted photons which are detected is $30.7 \times 0.034 \approx 1$.

If the apparatus were re-aligned using a dedicated 810nm alignment laser, instead of the current 780nm alignment laser, the optical attenuation factor of the free space portion of the apparatus could be improved by an order of magnitude. This would allow Alice to significantly reduce the number of photons transmitted per window: instead of over thirty, Alice could achieve a similar transmission efficiency by transmitting only 2 photons per transmission event. The Information Insecurity factor under these conditions would be $I = 0.08$. From that point, security could be further improved either by lowering the laser power (with a corresponding loss of efficiency) or by improving the optical alignment (which would allow Alice to transmit fewer photons without sacrificing efficiency.)

CHAPTER 4:
Conclusion

In this thesis I have advocated for the inclusion of QKD in NC3 secure key distribution infrastructure. I showed that it is within the scope of NPS's capabilities and academic interests to investigate the inclusion of QKD in national defense applications.

I first outlined the utility of QKD in Chapter 1 and described its development arc and the current state of QKD technology. I concluded that it will be worthwhile to pursue the development of QKD for national defense applications, and I outlined a case where QKD might offer increased robustness for submarine NC3 operations.

In Chapter 2, I developed a framework for information security in regards to QKD, first from a theoretical standpoint and then from a practical standpoint. I showed that a QKD apparatus capable of conducting free space secure key distribution at useful speeds can be assembled using commercial, off the shelf technology.

In Chapter 3, I described my efforts to build a demonstration apparatus capable of conducting free space QKD in laboratory conditions. I outlined the functions of the various pieces of the apparatus, as well as the necessary calibrations required for its assembly, and displayed a sample of the output. I also pointed out various spots where further modification could improve the operation of the apparatus. Those modifications are summarized in Table 4.1.

4.1 Remaining Tasks

The QKD demonstration apparatus described in Section 3.1 was shown to be capable of encoding keys in quantum states and transmitting them from Station A to Station B. However, the apparatus is not yet ready for use in actual QKD research. Table 4.1 discusses several areas where the QKD apparatus could be improved, outlines how those improvements might be achieved, and states the status of implementing those improvements.

Task	Reason	Required Equipment	Status
Install Polarization Rotation Block B	Until Polarization Rotation Block B is installed, the apparatus cannot measure photons in the diagonal basis and thus cannot implement BB84 protocol.	HV switching chip	Part identified, ordered
Improve HV circuit switching time	Current HV switching times are on the order of hundreds of milliseconds. COTS HV switching circuits can do that same task in 10μs.	Dedicated Pockel Cell driver	Part identified, vendor contacted
Improve Optical Alignment	810nm biphoton generator output is too weak to use for proper optical alignment. Current alignment is based on a 780nm beam. Because of this, beam coupling to free space and back to fiber is very inefficient.	810nm laser	Part identified, ordered
Electronically separate Stations A and B	Apparatus testing range is currently limited due to the fact that all of the control equipment is colocated.	Microcontollers with sufficiently high clock speed for wireless synchronization	Parts on hand, not programmed
Eliminate the 4μs offset between detection window and transmission window	The offset time is very large compared to the size of the window. Due to this, the insecurity factor of the apparatus is currently greater than 1 (and eavesdropper can obtain the entire key without risk of detection).	None	Need to program the FPGA in Quartus to wait 4μs before opening the detection window
Install a detector at the idler beam output of PBS A	This will increase security by allowing Alice to tell how many photons she sent to Bob in a given transmission event (called "heralding.")	One additional SPCM	Part identified, not ordered

Table 4.1. Summary of the remaining tasks and required equipment to finish the QKD apparatus described in Section 3.1.

4.2 Outlook

Discussions about defense applications of QKD sit at the intersection of national defense strategy, quantum physics, cryptology, and communications engineering. One goal of this paper has been to tie these disparate fields of study together so that interested parties may begin to form a common framework in order to discuss and develop different aspects of QKD for defense applications. As such, an effort was made to treat each aspect of the discussion at a relatively introductory level. Because of this, many aspects of the discussion around QKD detail have been left out.

For example, the only QKD protocol discussed in this paper is BB84. In reality, there are many QKD protocols, tailored to suit various special use cases. Of particular interest is the Eckert 1991 (E91) protocol [5], wherein the quantum states used in QKD are generated at a third point and distributed to both Alice and Bob. If point C were, say, a satellite, this protocol could allow Alice and Bob to conduct QKD without being in line of sight to one another.

In addition to protocols, there are various technological options for the production, transmission, and detection of single photons. For example, instead of employing degenerate Type-II SPDC in a PPKTP crystal for single photon production, photons could be produced by a very weak laser of a wavelength which propagates more efficienctly over free space. As for photon detection, instead of Si-APD style detectors, Superconducting Nanowire Single Photon Detectors (SNSPDs) could be used: these detectors are much more efficient over a broader range of wavelengths, but require cooling to below 1°K to operate.

These options and their relative strengths and weaknesses and optimal use cases, as well as others, have been tested by other organizations, and are discussed at length in the literature (Gisin, 2002 [30]; Xu, 2020 [22]). It would be relatively straightforward for NPS to acquire different pieces of QKD-related equipment and to recreate those tests in defense-oriented contexts. Coordination has already begun between NPS's Quantum Sensing Lab and Space Systems lab to outfit the QKD demonstration apparatus with adaptive optics in order to investigate the effects of range and environment on its operation.

After the tasks listed in Table 4.1 are complete and the apparatus has been optimized for operation at range, it could then be used to demonstrate the defense applications discussed in Chapter 1. For example, the effectiveness of different protocols and technological options could be demonstrated through collaboration with NPS's Center for Autonomous Vehicle Research (CAVR). Station A could be mounted on an unmanned aerial vehicle (UAV) and Station B mounted on an unmanned underwater vehicle (UUV), and the performance of the apparatus could be investigated under various conditions of range, relative speed, and environmental factors. These assets already exist

at NPS, and the equipment to be acquired and tested is all available commercially and is relatively non-specialized (meaning that acquisition, installation, and testing would all be straightforward.) Testing and demonstration of defense applications could be accomplished with a small price tag ($50-$500K) within one year of effort.

APPENDIX A:
Introduction to Cryptology

This appendix is intended to act as a handy introductory reference on the topics of cryptology and secure key distribution. It is intended to give the reader just enough background in these subjects so that they can better understand the analysis of QKD presented in Chapter 2.

A.1 Overview

The central problem of cryptology is that a message originator wishes to transmit a message to a certain intended receiver, while keeping the contents of the message secret from any potential eavesdroppers. This section discusses various topics related to cryptology in order to set the stage for the comparison of various cryptological infrastructures in Section A.2. Section A.1.1 provides a quick reference for terms related to cryptology. Sections A.1.2 and A.1.3 discuss how the basic concepts of cryptology fit together in the effort to conduct secure communications.

A.1.1 Vocabulary

- **Cryptology**. Cryptology is the overarching field of study which includes *cryptography* and *cryptoanalysis*.
- **Cryptography**. Cryptography is the art of rendering messages unintelligible, or *encryption*.
- **Cryptoanalysis.** Cryptoanalysis is the art of breaking codes, or *decryption*.
- **Key.** A key is a string of characters representing some function which is applied to the characters in a message: An *enciphering* key renders a message into an unintelligible cryptogram, while a *deciphering* key translates a cryptogram back into the original message. The characters which compose a key are collectively called "key material." In order for an enciphering key to generate secure cryptograms, each character of the key must be generated individually and randomly. In addition, the key must be at least as long as the message it encrypts.
- **Information-theoretical Security.** Security that is never compromised, regardless of the technological or temporal resources of an attacker.
- **Computational Security.** Security that can be breached given sufficient technological and temporal resources, but which cannot be breached within the span of time for which the secure material is required to remain secure.
- **One-time Pad.** An information-theoretically secure key. An information-theoretically secure key loses its secure status after the first time it is used to decrypt a message. For this

reason, it is also called a one-time pad. The terms "one-time pad" and "secure key" are used interchangeably in this paper.

- **Cryptogram.** A cryptogram is the string of characters resulting from the use of a key to encrypt a message.
- **Key distribution protocol.** A key distribution protocol is the process by which parties coordinate the encryption and decryption of secure communications by sharing a secure key. A simple example of a key distribution protocol is a child's toy decoder ring, used to decode scrambled messages found on the side of a cereal box.
- **Alice, Bob, and Eve.** In cryptological literature, Alice is the canonical originator of messages. Bob is the intended recipient of Alice's messages, and Eve (the nefarious eavesdropper) attempts to illicitly gain information about those messages. Unless otherwise noted, it is assumed that Eve is technologically on par with Alice and Bob, and that she has all possible equipment and capabilities required for cryptoanalysis.

A.1.2 Cryptographical knowledge

Cryptographical knowledge is the set of probabilities associated with possible keys and messages which might be exchanged between Alice and Bob. An example of cryptographical knowledge is the linguistic system of a transmission: if Bob and/or Eve know that a message will be written in a certain language, then they have some amount of cryptological knowledge, because they know that a particular set of possible messages (those in the language spoken by Alice and Bob) has a relatively higher associated probability than those that are not.

Knowledge gained by meta-analysis and intelligence, such as in the example above, constitutes *a priori* knowledge. The security of any key distribution protocol is defined in terms of how Eve's *a priori* knowledge compares with her *a posteriori* knowledge, where *a posteriori* knowledge refers to the set of probabilities associated with particular keys and messages after some amount of eavesdropping.

A.1.3 Unicity and security

If cryptoanalysis conducted by Bob or Eve causes the probability that a cryptogram corresponds to a certain message to approach unity, that analysis is said to have achieved *unicity*. That is, the holder of the cryptogram knows the precise contents of the message, and the code is "broken." Each key has a characteristic number, the *unicity distance*, which measures the amount of material which must be received by Bob or intercepted by Eve before unicity can be achieved.

If a key's unicity distance is infinite—i.e., Eve's *a posteriori* knowledge always matches her *a*

priori knowledge exactly, regardless of the volume of intercepted material and Eve's temporal or computational resources—then the cryptographic system is information-theoretically secure.[10] If the unicity distance is finite, but so large that Eve cannot achieve unicity while the secrecy of the message is still important, the system is computationally secure.

A.2 Secure Key Distribution

Two common categories of secure key distribution are SKI and public key infrastructure (PKI). They are described here in order to contrast them with QKD in terms of transmission security and speed.[11]

A.2.1 Symmetrical/Secret Key Infrastructure (SKI)

SKI requires a secure line of communication from Alice to Bob, which Alice uses to transmit key material. As long as Alice's key material is randomly generated and of sufficient length—at least equal to the length of the message—any cryptogram created from it can be shown to have "information-theoretic" security, which simply means that there is no way for a codebreaker to gain any information from it, regardless of temporal or computational resources. Therefore, after generating a secure key and transmitting it to Bob via the secure channel, Alice can use the key to create cryptograms and transmit them to Bob via whatever communication channel she chooses, regardless of the security of the channel.

While Shannon was able to prove SKI to be unconditionally secure [31], the key length requirement means that the speed of any information exchange is limited to the speed of the secure communication channel. High-volume lines of communication are typically vulnerable to infiltration by Eve, forcing key distribution to be conducted via secure low-volume channels. Typically, this involves the transfer of physical media, such as disks or codebooks. Thus, the security advantage of SKI is gained at the expense of speed.

A.2.2 Asymmetrical/Public Key Infrastructure (PKI)

PKI utilizes two keys: a *deciphering key* created by and known only to Bob, and an *enciphering key* which Bob calculates from his deciphering key via a "one-way function"—a function which

[10] In Shannon's seminal 1949 paper on the subject of cryptology, *Communication theory of secrecy systems* [31], Shannon introduced this same concept as "perfect secrecy."

[11] Note that security and speed are qualitative descriptors, defined situationally in relation to the needs of Alice and Bob. Unless explicitly defined, *secure* and *fast* should be understood to mean *secure enough* and *fast enough* to meet the needs of Alice and Bob, commensurate to the urgency and sensitivity of the information they are attempting to communicate.

cannot be reversed in quadratic time. Bob then makes the enciphering key publicly available so that Alice can access it use it to create a cryptogram. Alice can then transmit the cryptogram to Bob via whatever line of communication is convenient, knowing that only Bob can decipher it.

The first published PKI protocol was suggested by Diffie and Hellman in 1976 in a paper titled "New Directions in Cryptography" [32], but no one-way function was put forth at the time for its implementation. In 1983, Rivest et al. published the Rivest-Shamir-Adleman key exchange (RSA) protocol [33], suggesting prime factorization as a suitable one-way function: it is computationally easy for Bob to choose a set of prime numbers to use as a deciphering key, and it is also easy for him to calculate an enciphering key from that set of numbers (for example, the deciphering key could be the product of Bob's set of prime numbers.) At the time that RSA was first published, prime factorization was a computationally hard problem. A numerical demonstration of this principle is presented in Appendix B.

While not unconditionally secure, PKI is computationally secure, and it is significantly faster than SKI because it completely sidesteps the need for a secure communication channel.[12]

A.2.3 Comparison to Quantum Key Distribution

In short: SKI is theoretically secure, but it relies on the movement of physical media and therefore can be too slow or unwieldy for operational needs, while PKI is fast, but there is some uncertainty surrounding the level of security it provides. In comparison, QKD is just as secure as SKI, but it does not rely on physical media: as long as Alice and Bob have a line of sight to one another, they can generate information-theoretically secure key material at a distance via QKD. It will be seen in Chapter 2 that QKD is considerably slower than PKI, but its speed is adequate for the applications discussed in Section 1.2.

[12]This is only true as long as Eve is restricted to classical computers in her attacks. PKI is potentially vulnerable to attack by quantum computers using Shor's Algorithm [34]. In fact, the existence of "one-way functions" is yet to be proven, and as such any PKI protocol is potentially insecure.

APPENDIX B:
Public Key Infrastructure Example

PKI was briefly introduced in section A.2. This section continues the discussion and clarifies the idea of computational security by demonstrating how PKI, employed correctly, cannot be hacked in any realistic length of time by classical computers.

Treasure hunt scenario

Suppose Bob and Eve are racing to recover a buried treasure. Alice knows the coordinates of the treasure and decides to help Bob by telling him where to find it, but she knows that Eve is eavesdropping so she must communicate securely. Bob decides to facilitate the information exchange by creating an enciphering key 155 characters long in base 10 bits (equivalent to 512 binary bits). He chooses two prime numbers of sufficient length and multiplies them together to obtain an enciphering key. Perhaps he chooses the following seventy-eight digit numbers:

10263959282974110577205419657399167590071656780803806680334193352179071130777910660348838016845482092722036001287867920795857598929152227060823719306280864 3

He tells Alice the number he has chosen as his enciphering key, perhaps via a public tweet. Alice then accesses the enciphering key and uses it along with an encryption alorigthm to turn the coordinates of the treasure into a cryptogram. She then sends the cryptogram to Bob via a public tweet. Now, both Bob and Eve have the cryptogram which contains the encoded coordinates of the treasure, and they also both have the enciphering key. Bob, however, retains sole access to the deciphering key.

Armed with these pieces of information, Bob and Eve both take out their smart phones. While Bob is typing the deciphered coordinates into Google Maps, Eve navigates to Amazon's cloud computing service, Amazon EC2, and uses it to run Luke Valenta's Factoring as a Service [35] program, which she obtained from GitHub, in order to factor the enciphering key and obtain the deciphering key. Approximately four hours (and seventy five dollars) later, the program produces the deciphering key, and Eve quickly obtains the treasure's coordinates. Of course, Bob had the deciphering key already, and at this point he has a four hour head start on Eve.

Actual security of PKI

Bob's encryption scheme, known as RSA-155 (after the 155 base 10 digits of the enciphering key) was the standard for PKI until the late twentieth century. Using the factoring tools available at its initial adoption, hacking RSA-155 would have required many millions of years of computing time. But by 1999, computing power and factoring algorithms had advanced so far that a team was able to link several hundred supercomputers around the world and factor RSA-155 over a period of seven months [36].[13] Subsequent advances in computing power have resulted in the factoring of even larger numbers, the largest being RSA-768 which was factored in 2009 after two years of calculations [37]. RSA-155 is now know to be insufficiently secure for most applications.

Despite these successes, PKI can still be made computationally secure. For example, as demonstrated above, RSA-155 might be considered sufficiently secure if a certain piece of information needs to be secure for only a short period of time. Longer-term security can be achieved by using longer keys. Barring the surprise creation of an efficient factoring algorithm, PKI based on prime factorization will likely remain secure and useful until the advent of quantum computing (when this occurs, any feasible RSA protocol will be rendered insecure in a matter of minutes or hours by the employment of Shor's Algorithm [34]).

[13] This was done in response to a challenge issued by RSA Laboratories in 1991. A series of progressively larger RSA numbers were announced, with cash prizes promised to the teams who could factor them. The numbers used in the example above were the actual prime factors found by the team for the RSA-155 number given in the challenge.

APPENDIX C:
QKD Demonstration Apparatus Miscellaneous Pictures and Screenshots

Selected pictures and screenshots are included here to provide context to the explanations in Chapter 3.

C.1 QKD Demonstration Apparatus: Optical Portion

Photos of the optical portion of the QKD demonstration apparatus. Major components are labeled; minor optical components such as mirrors and static waveplates are shown but not labeled.

C.1.1 QKD Demonstration Apparatus: All Components

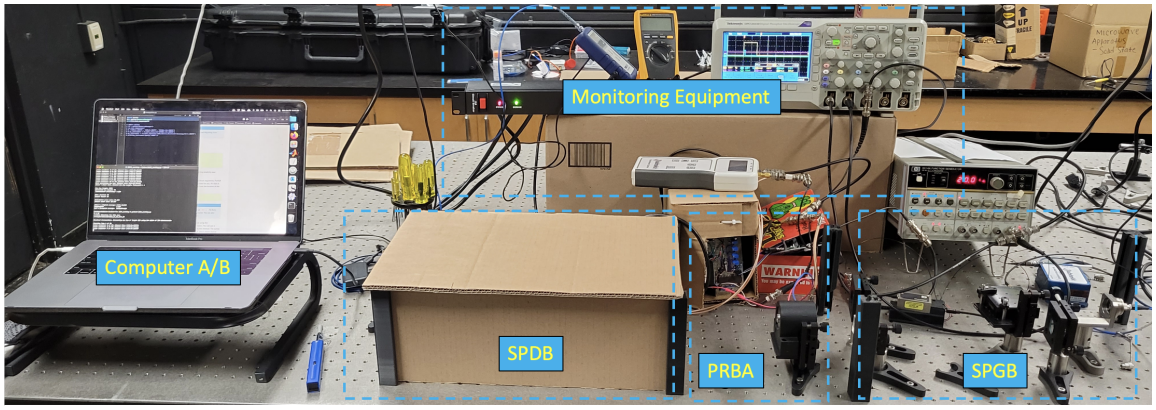

Figure C.1. Picture of the full QKD demonstration apparatus described in Section 3.1.

C.2 QKD Demonstration Apparatus: Single Photon Generation Block

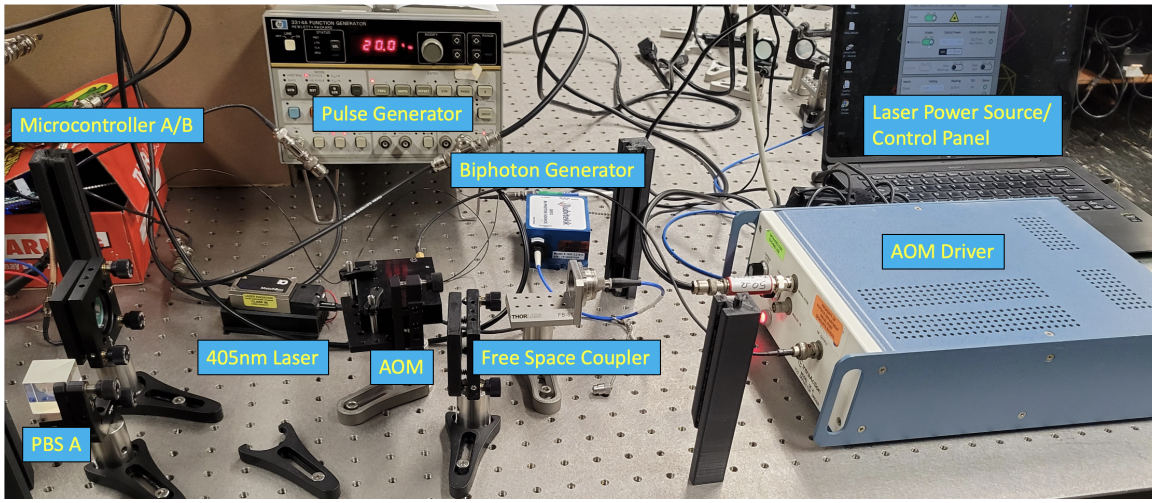

Figure C.2. Picture of the Single Photon Generation Block, described in Section 3.1.1.

C.3 QKD Demonstration Apparatus: Polarization Rotation Block

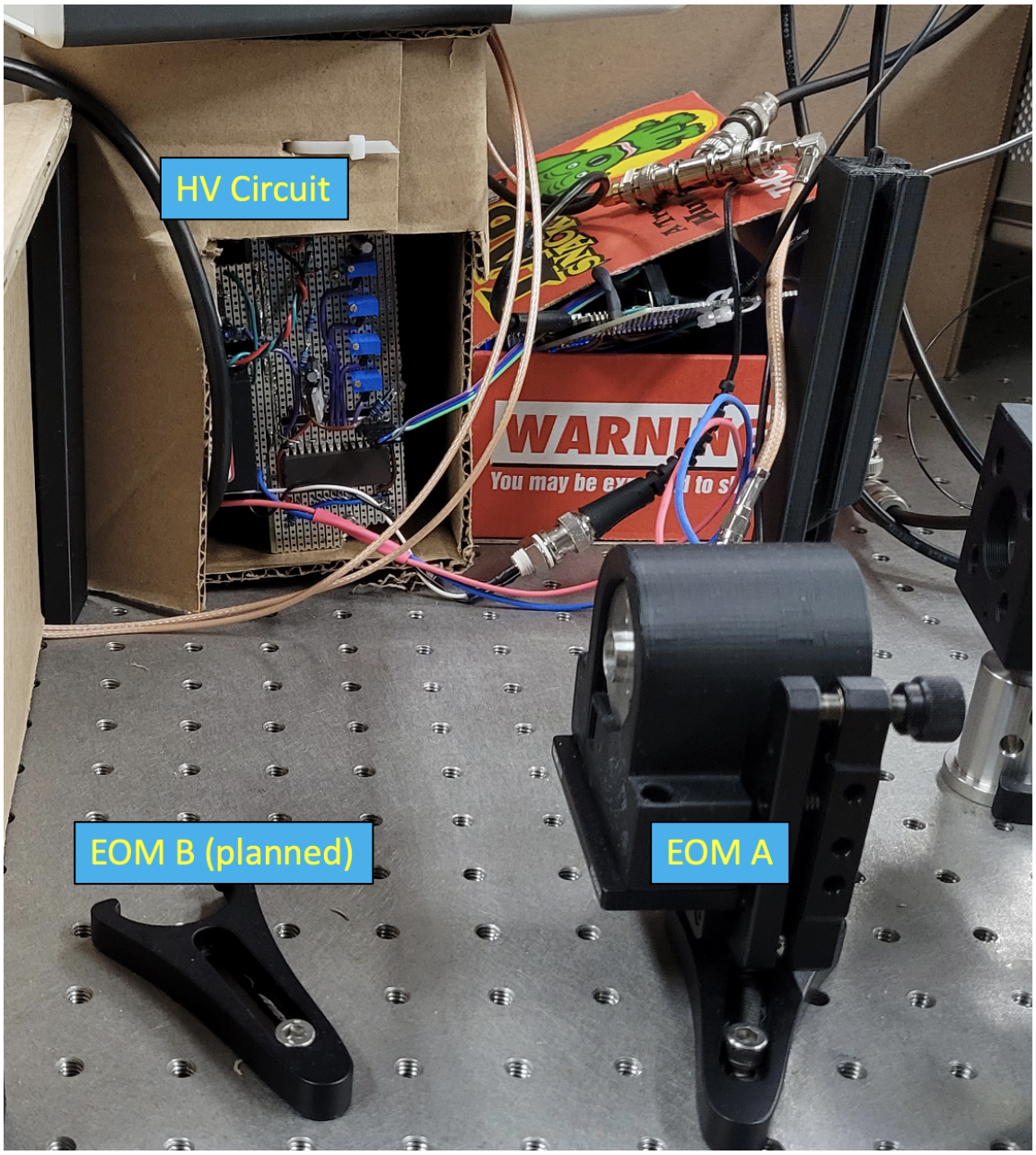

Figure C.3. Picture of Polarization Rotation Block A, described in Section 3.1.1.

C.4 QKD Demonstration Apparatus: Single Photon Detection Block

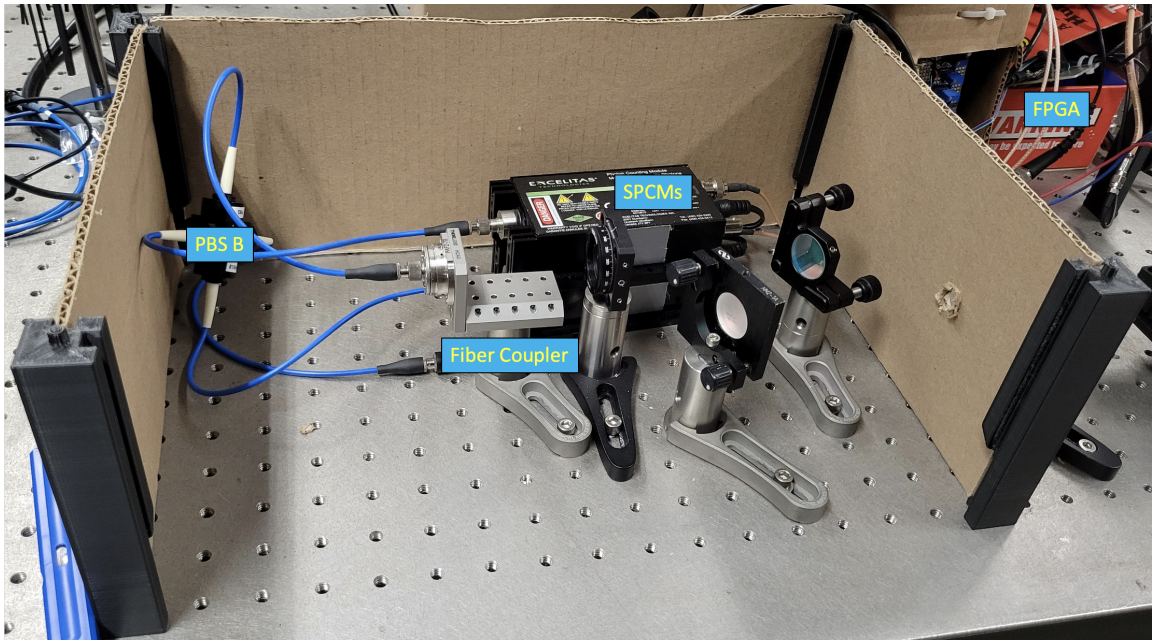

Figure C.4. Picture of the Single Photon Detection Block, described in Section 3.1.2.

C.5 QKD Demonstration Apparatus: Software Portion

The scripts controlling various aspects of the QKD demonstration apparatus, along with brief descriptions of their functionality.

C.5.1 QKD Demonstration Apparatus: Computer A

The script on Computer A is written in Python 3. It communicates with Microcontroller A via PySerial. User enters a message to be encrypted into a text file and enters the filepath of the message in the "Retrieve message from text file" section. User enters desired transmission parameters into the section labeled "User Inputs". When run, the script coordinates with Microcontoller A to generate lists of randomly generated raw key bits and polarization bases. When key generation is complete, the script retrieves those lists and stores them in a new folder. User enters the desired filepath of this new folder in the "Setup" section, under "Create data directory". Subsequently, the script processes the retrieved information in order to implement BB84 protocol and determine usability of the generated key.

```
# This script is for Computer A of the BB84 Quantum Key Distribution (QKD) demonstration
# apparatus.

# Alice records a message in a text file.
#
# This script reads that message and then generates a secure key by interacting with an
# optical setup via an Arduino plugged into the computer's USB port.
#
# This script then processes the key and generates a version of the key for Alice and a
# version of the key for Bob.
#
# Then, the script calculates and displays the error rate between the two keys.
#
# Finally, it uses the keys to encrypt and then decrypt Alice's message and displays the
# results.

# ==============================
# Libraries
# ==============================

import os
import math
import time
import serial
import random
import numpy as np
from   datetime import datetime

# ==============================
# Retrieve message from text file
# ==============================

messageFile = open( 'messageToSend/messageToSend.txt', 'r' )
message     = messageFile.read()
messageFile.close()

# message = "x" # Un-comment if no message is to be transmitted
```

```
# ==============================
# User inputs
# ==============================

keyLength = 10 # Number of raw bits to generate per loop (don't exceed 5000)
repLength = 10 # Artifact of previous editions of code - should be set equal to keyLength
n         = 1  # Number of loops (total length of raw key = keyLength * n)

binSize    = 20.0 # frequency as read on the pulse generator (kHz)

cryptNum   = 3 # for cipher: how many bits to devote to scrambling each character

# ==============================
# Boring stuff
# ==============================

# Variable initialization
# ==============================
testing    = []
lenCheck   = []
Ch1        = []
Ch2        = []
TB         = []
BA         = []
BB         = []
jitter     = 300

# Conversion of pulse frequency to pulse length
# ==============================
bucketSize = ( 1 / binSize ) * 500 + jitter # Change Hz to microseconds and add a buffer
                                            # for Arduino clock jitter

# Variables used for testing/debugging
# ==============================
# testing    = 1 # un-comment to set Bob's key equal to Alice's for testing/debugging
lenCheck   = 0 # Comment to turn on length check

# ==============================
```

```
# Setup
# ==============================


# Create data directory
# ==============================
DTG  = datetime.today().strftime( '%Y_%m_%d_%H%M_%S' )
path = 'keyGen/07Jul06/%s' % DTG
os.mkdir( path )
print( '***Data directory for this run:\n***%s' % path )
dataFile = open( '%s/counts.txt' % path, 'a' )
keyFile = open( '%s/key.txt' % path, 'a' )

# Open serial port
# ==============================

ser = serial.Serial(

port = '/dev/cu.usbmodem141101',
baudrate = '19200'

)

# ============================================================
# ============================================================

# Do the thing

# ============================================================
# ============================================================

# Start counting at the beginning of the next whole second
# ==============================

T = float( datetime.today().strftime( '%S.%f' )[:-6] )

while( T > 1 ):
```

```
    T = T - 1

# Main loop
# ==============================

t0 = datetime.now()

for g in range( 0, n ):

# Send the start command
ser.write( b'<%0d,%0d,%0d>\n' % ( T, repLength, bucketSize ) )
ser.flush()

if( g == 0 ):

# Print the Arduino's acknowledgement for the first loop
# ==============================
print( "\n" + str( ser.readline() ).lstrip( 'b\'' ).rstrip( '\\r\\n\'' ) )
print( "\nStarting loop 1 of " + str( n ) )

else:

ser.readline() # Disregard the Arduino's ready message on subsequent loops

# Retrieve counts for this loop
# ==============================
x = ( str( ser.readline() ).lstrip( 'b\'' ).rstrip( '\\r\\n\'' ) )
countsCh1, countsCh2, trueBits, basisAlice, basisBob = x.split( ';' )

countsCh1  = countsCh1.rstrip( "," ).split( "," )
countsCh2  = countsCh2.rstrip( "," ).split( "," )
trueBits   = trueBits.rstrip( "," ).split( "," )
basisAlice = basisAlice.rstrip( "," ).split( "," )
basisBob   = basisBob.rstrip( "," ).split( "," )

# Store the counts in a data file
# ==============================
for i in range( 0, len( countsCh1 ) ):
```

```
   dataFile.write( '%s, %s\n' % ( countsCh1[ i ], countsCh2[ i ] ) )
   keyFile.write( '%s, %s, %s\n' % ( trueBits[ i ], basisAlice[ i ], basisBob[ i ] ) )

# Save all retrieved info into variables for later use
# ==============================
Ch1 = Ch1  + countsCh1
Ch2 = Ch2  + countsCh2
TB  = TB   + trueBits
BA  = BA   + basisAlice
BB  = BB   + basisBob

timeLoop = datetime.now() - t0 # Loop end time
print( "Loops complete: " + str( g + 1 ) + " of " + str( n ) + "; elapsed time: "
+ str( timeLoop ) ) # Show completed loop index

if( g < n - 1 ):

ser.readline() # Disregard the Arduino's end message when the key isn't done

else:

tf = datetime.now() - t0
print("Done generating ray key. Elapsed time: " + str( tf ) )
print(str(ser.readline()).lstrip('b\'').rstrip('\\r\\n\'')) # Print end message

# Close out
# ==============================
dataFile.close()
keyFile.close()

# ============================================================
# ============================================================

# Processing

# ============================================================
# ============================================================
```

```
# ==============================
# Boring stuff
# ==============================

countsCh1      = Ch1
countsCh2      = Ch2
trueBits       = TB
basisAlice     = BA
basisBob       = BB
bobsKey        = []
badBits        = []
alisKey        = []
checkBob       = []
checkAli       = []
bobsCheckedKey = []
alisCheckedKey = []

errorsFound    = 0
errorsActual   = 0

Akeydisp = []
Bkeydisp = []

# ==============================
# Sifting step
# ==============================

for i in range( 0, len( trueBits ) ):

if( int( basisBob[ i ] ) == int( basisAlice[ i ] ) and int( countsCh1[ i ] ) !=
int( countsCh2[ i ] ) and int( countsCh2[ i ] ) <= 1 and int( countsCh1[ i ] ) <= 1 ):

bobsKey.append( int( countsCh1[ i ] ) ) # Bob constructs his side of the key
badBits.append( 0 )
Bkeydisp.append( int(countsCh1[ i ]) )
Akeydisp.append( int(trueBits[ i ]) )
```

```
else:

badBits.append( 2 ) # Bob tells Alice which bits to throw out
Bkeydisp.append( 2 )
Akeydisp.append( 2 )

for i in range( 0, len( trueBits) ):

if( badBits[ i ] == 0 ):

alisKey.append( int( trueBits[ i ] ) ) # Alice constructs her side of the key

# ==============================
# Comparison step
# ==============================

for i in range( 0, len( bobsKey ) ):

chooseBit = random.randrange( 2 )

if( chooseBit == 1 ):

checkBob.append( bobsKey[ i ] )
checkAli.append( alisKey[ i ] )

else:

bobsCheckedKey.append( bobsKey[ i ] )
alisCheckedKey.append( alisKey[ i ] )

for i in range( 0, len( checkBob ) ):

if( checkBob[ i ] != checkAli[ i ] ):

errorsFound = errorsFound + 1

for i in range( 0, len( bobsCheckedKey ) ):
```

```
if( bobsCheckedKey[ i ] != alisCheckedKey[ i ] ):

errorsActual = errorsActual + 1

# ============================
# If error rate is low enough, record the key and move on. If not, try again.
# ============================

if( len( bobsCheckedKey ) >= len( message )  * 8 or lenCheck == 0 ):

errorRateCalculated    = float( errorsFound )/len( checkBob )*100
errorRateActual        = float( errorsActual )/len( bobsCheckedKey )*100
transmissionEfficiency = float( len( bobsCheckedKey ) ) / ( keyLength * n ) * 100

print( "\nRaw key length: "              + str( len( countsCh1 ) )      )
print( "Generated key length: "          + str( len( bobsCheckedKey ) ) )
print( "Transmission efficiency: %.2f%%" % transmissionEfficiency       )
print( "\nBits checked: "                + str( len( checkBob ) )       )
print( "Errors found: "                  + str( errorsFound )           )
print( "\nCalculated error rate: %.2f%%" % errorRateCalculated          )
print( "Actual error rate: %.2f%%"       % errorRateActual              )

keyFacts = open('%s/keyFacts.txt' % path , 'a' )

keyFacts.write( "\nRaw key length:        %s\n"     % str( len( countsCh1 ) )      )
keyFacts.write( "Generated key length:    %s\n"     % str( len( bobsCheckedKey ) ) )
keyFacts.write( "Transmission efficiency: %.2f%%\n" % transmissionEfficiency       )
keyFacts.write( "\nBits checked:          %s\n"     % str( len( checkBob ) )       )
keyFacts.write( "Errors found:            %s\n"     % str( errorsFound )           )
keyFacts.write( "Calculated error rate:   %.2f%%\n" % errorRateCalculated         )
keyFacts.write( "\nActual error rate:     %.2f%%"   % errorRateActual              )

keyFacts.close()

finalKeyBob = open( '%s/bobsKey.txt' % path, 'a' )
finalKeyAli = open( '%s/alisKey.txt' % path, 'a' )

for i in range( 0, len( bobsCheckedKey ) ):
```

```
finalKeyBob.write( '%s' % str( bobsCheckedKey[ i ] ) )
finalKeyAli.write( '%s' % str( alisCheckedKey[ i ] ) )

finalKeyBob.close()
finalKeyAli.close()

else:

print("Your key was too short! You only generated " + str( len( bobsCheckedKey ) )
 + " characters (you needed " + str( len( message ) * 8 ) + "). Try again.")

# ============================================================
# ============================================================

# Cryptography

# ============================================================
# ============================================================

# ==============================
# Boring stuff
# ==============================

decipher         = []
cryptogram       = []
cryptogramAscii  = []
cipher           = []
messageAscii     = []
decrypt          = []
bobsMessage      = []

asciiTableLength = 255
asciiMin         = 32

# ==============================
# Retrieve the keys from the files they were just saved to
# ==============================
```

```
alice  = open( '%s/alisKey.txt' % path, 'r' )
keyAli = alice.read()
alice.close()

bobby  = open( '%s/bobsKey.txt' % path, 'r' )
keyBob = bobby.read()
bobby.close()

# ==============================
# Make a cryptogram using Alice's key
# ==============================

for j in range( 0, len( message ) ):

cipher.append( int( keyAli[ j * cryptNum : j * cryptNum + cryptNum ], base = 2 ) )
cryptogramAscii.append( ord( message[ j ] ) + cipher[ j ] + asciiMin )

if( cryptogramAscii[ j ] > asciiTableLength ):

cryptogramAscii[ j ] = cryptogramAscii[ j ] - asciiTableLength + asciiMin

cryptogram.append( chr( cryptogramAscii[ j ] ) )

# ==============================
# ==============================

# Transmit the cryptogram along a public channel (just pretending for now)

# ==============================
# ==============================

# ==============================
# Use Bob's key to decipher the cryptogram
# ==============================

if( testing == 1 ):
```

```
keyBob = keyAli

for j in range( 0, len( message ) ):

decipher.append( int( keyBob[ j * cryptNum : j * cryptNum + cryptNum ], base = 2 ) )
decrypt.append( ord( cryptogram[ j ] ) - decipher[ j ] - asciiMin)

if( decrypt[ j ] <= asciiMin ):

decrypt[ j ] = decrypt[ j ] + asciiTableLength - asciiMin

if( decrypt[ j ] >= asciiMin and decrypt[ j ] != 255 ):

bobsMessage.append( chr( decrypt[ j ] ) )

else:

bobsMessage.append( " " )

# ==============================
# See how you did!
# ==============================

print( "\nAlice's original message: " + message )
print( "Cryptogram:               " + "".join( cryptogram ) )
print( "Bob's decrypted message:  " + "".join( bobsMessage ) + "\n")

for i in range( 0, len( Ch1 ) ):

Ch1[ i ] = int( Ch1[ i ] )
Ch2[ i ] = int( Ch2[ i ] )
TB[ i ] = int( TB[ i ] )
BA[ i ] = int( BA[ i ] )
BB[ i ] = int( BB[ i ] )

# Uncomment below to see results in the terminal box
# ==============================
# print( Ch1 )
```

```
# print( Ch2 )
# print( BA )
# print( BB )
# print( TB )
# print( Bkeydisp )
# print( Akeydisp )
# print( bobsCheckedKey )
# print( alisCheckedKey )
```

C.5.2 QKD Demonstration Apparatus: Microcontroller A

The script on Microcontroller A is written in C++. It interacts with Computer A via the Streaming library, available through the free downloadable Arduino application. Upon power-up, the script defines variables, establishes connection with Computer A, and sets TTL pin voltage levels. While no input is available via serial USB input from Computer A, Microcontroller A continuously loops the function getParameters(). When an input is available, the parseData() function reads it and extracts commands. Parsed commands are then sent to the function doTheThing(), which generates random raw key bits and polarization bases and coordinates with the optical apparatus to conduct key transmission. When transmission is complete, the endTransmission() function sends the lists of bits, bases, and transmitted data back to Computer A and resets variables and pins in order to prepare for another transmission.

```
/*

This script is for Microcontroller A of the BB84 Quantum Key Distribution (QKD) demonstration a

It takes inputs from a controlling station via serial USB, then generates a raw key by
periodically pulsing a BB84 optical setup. It also generates and stores the transmitted
bits, the bases chosen by the sender and receiver (Alice and Bob), and the raw key
generated at the receiver end. When the raw key is complete, all of the stored data is
transferred to the controlling station and the Arduino resets itself.

 updated by Jack Brault on 7 April 2021

*/

// ============================
```

```
// Libraries
// ==============================

#include<Streaming.h>

// ==============================
// Definitions
// ==============================

// Stuff for communicating with the PC
// ==============================
const byte buffSize = 40;
const char startMarker = '<';
const char endMarker = '>';
boolean readInProgress = false;
char inputBuffer[buffSize];
byte bytesRecvd = 0;

// Pin names
// ==============================
int pinCounts[]            = { D2, D3, D4, D5, D6, D7, D8, D9 };
int pinOverflow            = D10;
int pinClr                 = D11;
int pinGateEn              = D12;
int pinGatePulse           = D13;
int pinChannelChooser[]    = { A0, A1 };
int pinWin[]               = { A2, A3, A4 };
int pinBasisBob            = A5;
int pinBitAlice            = A6;
int pinBasisAlice          = A7;

// Serial inputs
// ==============================
int keyLengthRequested   = 5000;
int pulseWidth              = 0;

// Stored info
// ==============================
```

```
int countsIn              [ 8 ];
int countsDeci      [2][ 5000 ];
int trueBits           [ 5000 ];
int basisAlice         [ 5000 ];
int basisBob           [ 5000 ];
int del = 260;

// Counting/loop indices
// ==============================
int i                       = 0;
int l                       = 0;
int keyLength               = 0;
int pinSetting              = 0;

// Time keeping
// ==============================
unsigned long timeRecv      = 0;
unsigned long timeStart     = 0;
unsigned long binTime       = 0;

// Flags
// ==============================
boolean readyMessage    = false;  // Displays format for input entry
boolean countsOverflow  = false;  // Informs that an overflow has occurred

// ==============================
// Setup
// ==============================

void setup() {

  serialBootup();                 // Initializes serial communications
  pinDefs();                      // Initializes pins

}

// ==============================
```

```
void serialBootup() {

  Serial.begin( 19200 );
  while ( !Serial ) {;}

}

// ==============================
// Assign all pins
// ==============================

void pinDefs() {

  pinMode( pinCounts[ 0 ],            INPUT );
  pinMode( pinCounts[ 1 ],            INPUT );
  pinMode( pinCounts[ 2 ],            INPUT );
  pinMode( pinCounts[ 3 ],            INPUT );
  pinMode( pinCounts[ 4 ],            INPUT );
  pinMode( pinCounts[ 5 ],            INPUT );
  pinMode( pinCounts[ 6 ],            INPUT );
  pinMode( pinCounts[ 7 ],            INPUT );
  pinMode( pinOverflow,               INPUT );
  pinMode( pinClr,                   OUTPUT );
  pinMode( pinGateEn,                OUTPUT );
  pinMode( pinGatePulse,             OUTPUT );
  pinMode( pinChannelChooser[ 0 ],   OUTPUT );
  pinMode( pinChannelChooser[ 1 ],   OUTPUT );
  pinMode( pinWin[ 0 ],              OUTPUT );
  pinMode( pinWin[ 1 ],              OUTPUT );
  pinMode( pinWin[ 2 ],              OUTPUT );
  pinMode( pinBasisBob,              OUTPUT );
  pinMode( pinBasisAlice,            OUTPUT );
  pinMode( pinBitAlice,              OUTPUT );
  digitalWrite( pinCounts[ 0 ],         LOW );
  digitalWrite( pinCounts[ 1 ],         LOW );
  digitalWrite( pinCounts[ 2 ],         LOW );
  digitalWrite( pinCounts[ 3 ],         LOW );
  digitalWrite( pinCounts[ 4 ],         LOW );
```

```
  digitalWrite( pinCounts[ 5 ],         LOW );
  digitalWrite( pinCounts[ 6 ],         LOW );
  digitalWrite( pinCounts[ 7 ],         LOW );
  digitalWrite( pinOverflow,            LOW );
  digitalWrite( pinClr,                 LOW );
  digitalWrite( pinGateEn,              LOW );
  digitalWrite( pinGatePulse,           LOW );
  digitalWrite( pinChannelChooser[ 0 ], LOW );
  digitalWrite( pinChannelChooser[ 1 ], LOW );
  digitalWrite( pinWin[ 0 ],            LOW );
  digitalWrite( pinWin[ 1 ],            LOW );
  digitalWrite( pinWin[ 2 ],            LOW );
  digitalWrite( pinBasisBob,            LOW );
  digitalWrite( pinBasisAlice,          LOW );
  digitalWrite( pinBitAlice,            LOW );

}

// ==============================

// ==============================
// Main loop
// ==============================

// ==============================
// ==============================

void loop() { getParameters(); }

// ==============================
// ==============================

void doTheThing() {

  digitalWrite( pinGateEn , HIGH ); // Enable key generation

  for ( int i = 1 ; i <= keyLengthRequested ; i++) {
```

```
    clrFPGA();
    gateOpenShut();
    readCounts();
    keyLength++;
    readyMessage = false;

  }

  digitalWrite( pinGateEn, LOW );
  endTransmission(); // Sends stored data via USB
                     // and resets the Arduino in preparation for another key

}

// ==============================
// ==============================

void parseData() {

  char * strtokIndx;

  strtokIndx = strtok( inputBuffer, "," );
  timeStart = atoi( strtokIndx );

  strtokIndx = strtok( NULL, "," );
  keyLengthRequested = atoi( strtokIndx );

  strtokIndx = strtok( NULL, "," );
  pulseWidth = atoi( strtokIndx );

  Serial << "Received instructions. Generating raw key of length " << keyLengthRequested
  << " using bin width of " << pulseWidth << " microseconds." << endl;

}

// ==============================
// ==============================
```

```
void clrFPGA() {

  digitalWrite( pinClr, HIGH );
  digitalWrite( pinClr, LOW );

}

// ==============================
// ==============================

void gateOpenShut() {

// Comment/uncomment/tweak individual lines/blocks according to desired function.

/*
   // If switching bases randomly
   // ==============================

    pinSetting = random( 2 );
    basisBob[ keyLength ] = pinSetting;
    if ( pinSetting == 0 ) { digitalWrite( pinBasisBob , LOW ); }
    else { digitalWrite( pinBitAlice , HIGH ); }

    pinSetting = random( 2 );
    basisAlice[ keyLength ] = pinSetting;
    if ( pinSetting == 0 ) { digitalWrite( pinBasisAlice , LOW ); }
    else { digitalWrite( pinBasisAlice, HIGH ); }

    pinSetting = digitalRead( pinBasisAlice );
    basisAlice[ keyLength ] = pinSetting;
    if( pinSetting == 0 ) { digitalWrite( pinBasisAlice, HIGH ); }
    if( pinSetting == 1 ) { digitalWrite( pinBasisAlice, LOW  ); }

    pinSetting = digitalRead( pinBasisBob );
    basisBob[ keyLength ] = pinSetting;
    if( pinSetting == 0 ) { digitalWrite( pinBasisBob, HIGH ); }
    if( pinSetting == 1 ) { digitalWrite( pinBasisBob, LOW  ); }
*/
```

```
// If no basis switching
    // ============================
    basisBob[ keyLength ] = 0;
    basisAlice[ keyLength ] = 0;

    // Generate key bits
    // ============================
    pinSetting = random( 2 ); // random key
//    pinSetting = 0;         // all zeros

    if( pinSetting == 0 ) { trueBits[ keyLength ] = 1; }
    if( pinSetting == 1 ) { trueBits[ keyLength ] = 0; }
    if( pinSetting == 0 ) { digitalWrite( pinBitAlice , HIGH ); }
    else { digitalWrite( pinBitAlice, LOW ); }

    // Open and close the counting gate
    // ============================
    delay( del );                              // Wait for the HV circuit to set
    binTime = micros();                        // Open time bin
    digitalWrite( pinGatePulse , HIGH );       // Open the gate
    while ( micros() < binTime + pulseWidth ) ; // Wait until gate closes
    digitalWrite( pinGatePulse , LOW );        // Close time bin
}

//============================
//============================

void readCounts() { for ( int i = 0 ; i <= 1 ; i++ ) {

    if ( i == 0 ) { digitalWrite( pinChannelChooser[ 0 ] , HIGH ); }
    else { digitalWrite( pinChannelChooser[ 1 ] , HIGH ); }

    if ( digitalRead( pinOverflow ) == 1 ) { countsOverflow = true; }

    if ( countsOverflow )  {

      for ( int j = 0 ;  j <= 7; j++ ) { countsIn[ j ] = 0; }
```

```
      Serial << "Overflow detected on bit " << keyLength+1 << " of channel " << i
      << ". Bit set to zero." << endl;
      countsOverflow = false;

    }

    else { for ( int j = 0 ; j <= 7 ; j++ )
    { countsIn[ j ] = digitalRead( pinCounts[ j ] ); } }

    if ( i == 0 ) { digitalWrite( pinChannelChooser[ 0 ] , LOW ); }
    else { digitalWrite( pinChannelChooser[ 1 ] , LOW ); }

    int l = 0;
    for ( int k = 0 ; k <= 7 ; k++ ) { l = l + countsIn[ k ]*pow( 2, k ); }

    countsDeci [ i ][ keyLength ] = l;

  }
}

//==============================
//==============================

void endTransmission() {

    for ( int i = 0 ; i <= keyLengthRequested - 1 ; i++ )
    { Serial.print( countsDeci[ 0 ][ i ] ); Serial.print( "," ); }
    Serial.print( ";" );
    for ( int i = 0 ; i <= keyLengthRequested - 1 ; i++ )
    { Serial.print( countsDeci[ 1 ][ i ] ); Serial.print( "," ); }
    Serial.print( ";" );
    for ( int i = 0 ; i <= keyLengthRequested - 1 ; i++ )
    { Serial.print( trueBits[ i ] ); Serial.print( "," ); }
    Serial.print( ";" );
    for ( int i = 0 ; i <= keyLengthRequested - 1 ; i++ )
    { Serial.print( basisAlice[ i ] ); Serial.print( "," ); }
    Serial.print( ";" );
    for ( int i = 0 ; i <= keyLengthRequested - 1 ; i++ )
```

```
    { Serial.print( basisBob[ i ] ); Serial.print( "," ); }
    Serial.println();

    for ( int i = 0 ; i <= 5000 ; i++ ) {

      countsDeci [ 0 ][ i ] = 0;
      countsDeci [ 1 ][ i ] = 0;
      trueBits   [ i ]      = 0;
      basisAlice [ i ]      = 0;
      basisBob   [ i ]      = 0;

      }

    keyLength = 0;

}

//==============================
//==============================

void getParameters() {

  if ( !readyMessage ) {
  Serial << "Enter < Start time, Desired Raw Key Length, Pulsewidth > :" << endl;
  readyMessage = true; }

  while ( Serial.available() < 1 ) ; // Do nothing until a serial input is received

  char x = Serial.read();
  timeRecv = micros();

  if ( x == endMarker ) {

    readInProgress = false;
    inputBuffer[bytesRecvd] = 0;
    parseData();

    while ( micros() < timeRecv + timeStart ) ; // Wait until prescribed start time
```

```
    doTheThing();

  }

  if ( readInProgress ) {

    inputBuffer[bytesRecvd] = x;
    bytesRecvd ++;
    if ( bytesRecvd == buffSize ) { bytesRecvd = buffSize - 1; }

  }

  if ( x == startMarker ) {

    bytesRecvd = 0;
    readInProgress = true;

  }
}

//=============================
//=============================
```

C.5.3 QKD Demonstration Apparatus: Field Programmable Gate Array

The FPGA is programmed in Quartus II, a Hardware Description Language (HDL) from Intel. The bulk of the program was developed by Qubitekk for its CC1 Coincidence Counter, but was tweaked for this application in order to make it loop more quickly. The program begins counting pulses on its input pins when a gate signal is received, and stops when the gate signal is removed. When counts exceed 255, an Overflow bit is recorded. After counts are recorded, Microcontroller A prompts the FPGA to send its recorded data via TTL serial connection. Once Microcontroller A receives the data, it prompts the FPGA to clear and reset in preparation for the next gate signal.

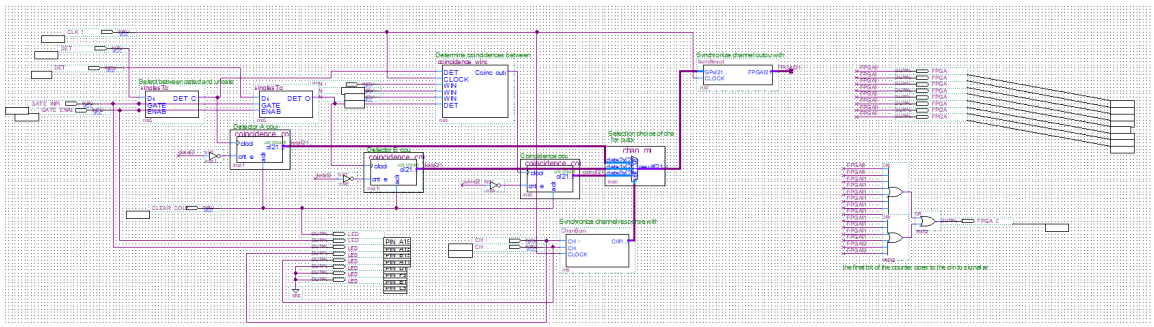

Figure C.5. Snippet of the control script on the FPGA described in Section 3.1.2. This HDL script detects pulses from the SPCMs and passes information to Microcontroller B. Written in Quartus II.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] R. Uppal. (2020). Military developing free space optical (FSO) or laser communications for ultrafast secure communications that are harder to detect and disrupt. [Online]. Available: InternationDefense,Security&Technology.https://idstch.com/technology/photonics/military-developing-free-space-opticalfso-or-laser-communications-for-ultrafast-secure-communications-that-are-harder-to-detect-and-disrupt/. Accessed 22 Jan 2021.

[2] S. J. Wiesner, "Conjugate coding," *Computer Science*, 1983.

[3] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," in *International Conference on Computers, Systems & Signal Processing*, 1984, vol. 1 of 3, pp. 175–179.

[4] C. H. Bennett and G. Brassard, "Experimental quantum cryptography: The dawn of a new era for quantum cryptography: The experimental prototype is working," *SIGACT News*, vol. 20, no. 4, p. 78–80, Nov. 1989. [Online]. Available: https://doi.org/10.1145/74074.74087

[5] A. K. Ekert, "Quantum cryptography based on Bell's theorem," in *Physical Review Letters*, 1991, vol. 67, pp. 661–663.

[6] M. Christandl, R. Renner, and A. Ekert. (2004). A Generic Security Proof for Quantum Key Distribution. [Online]. Available: eprintarchive. Http://arxiv.org/abs/quant-ph/0402131.

[7] C. Elliot and H. Yeh, "DARPA QUANTUM NETWORK TESTBED," Defense Advanced Research Projects Agency, VA, Tech. Rep. AFRL-IF-RS-TR-2007-180, July 2007.

[8] R. Pease. (2008). 'Unbreakable' encryption unveiled. [Online]. Available: http://news.bbc.co.uk/2/hi/science/nature/7661311.stm. Accessed 18 Dec 2020.

[9] F. Xu, W. Chen, S. Wang, Z. Yin, Y. Zhang, Y. Liu, Z. Zhou, Y. Zhao, H. Li, D. Liu, Z. Han, and G. Guo, "Field experiment on a robust hierarchical metropolitan quantum cryptography network," 2009.

[10] M. Sasaki, M. Fujiwara, H. Ishizuka, W. Klaus, K. Wakui, M. Takeoka, S. Miki, T. Yamashita, Z. Wang, A. Tanaka, and et al. (2011, May). Field test of quantum key distribu-tion in the Tokyo QKD Network. *Optics Express*. [Online]. *19*(11). p. 10387. Available: http://dx.doi.org/10.1364/OE.19.010387

[11] Juan Yin et al., "Satellite-to-ground entanglement-based quantum key distribution," *Physical Review Letters*, vol. 119, no. 20, pp. 200 501–1–5, Nov. 2017.

[12] Shang-Kai Liao et al., "Satellite-relayed intercontinental quantum network," *Physical Review Letters*, vol. 120, no. 3, pp. 030 501–1–4, Jan. 2018.

[13] C. J. Pugh, S. Kaiser, J.-P. Bourgoin, J. Jin, N. Sultana, S. Agne, E. Anisimova, V. Makarov, E. Choi, B. L. Higgins, and et al. (2017, Jun). Airborne demonstration of a quantum key distribution receiver payload. *Quantum Science and Technology*. [Online]. *2*(2). p. 024009. Available: http://dx.doi.org/10.1088/2058-9565/aa701f

[14] L. Gasman. (2019). Inside Quantum Technology Report says Quantum Key Distribution Market to reach $980 million by 2024. [Online]. Available: https://www.globenewswire.com/news-release/2019/04/30/1812788/0/en/Inside-Quantum-Technology-Report-says-Quantum-Key-Distribution-Market-to-Reach-980-million-by-2024.html. Accessed 9 Jan 2021.

[15] ISRO makes breakthrough demonstration of free-space Quantum Key Distribution (QKD) over 300 m. (2021). [Online]. Available: https://www.isro.gov.in/update/22-mar-2021/isro-makes-breakthrough-demonstration-of-free-space-quantum-key-distribution-qkd. Accessed 22 Jun 2021.

[16] Quantum Key Distribution (QKD) and Quantum Cryptography (QC). (2020). [Online]. Available: https://www.nsa.gov/what-we-do/cybersecurity/quantum-key-distribution-qkd-and-quantum-cryptography-qc/. Accessed 8 Jan 2021.

[17] J. Prisco. (2020). To QKD Or Not To QKD: What Quantum Key Distribution Means For Business. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2020/12/18/to-qkd-or-not-to-qkd-what-quantum-key-distribution-means-for-business/?sh=6561022c3e1c. Accessed 9 Jan 2021.

[18] Department of Energy. (2019, Aug. 29). Department of Energy Announces $60.7 Million to Advance Quantum Computing and Networking. [Online]. Available: https://www.energy.gov/articles/department-energy-announces-607-million-advance-quantum-computing-and-networking

[19] Verizon achieves milestone in future-proofing data from hackers. (2020). [Online]. Available: https://www.verizon.com/about/news/verizon-achieves-milestone-future-proofing-data-hackers. Accessed 23 Jun 2021.

[20] J. Russell, "Application of quantum key distribution," in *MILCOM 2008 - 2008 IEEE Military Communications Conference*, 2008, vol. 1, pp. 1–6.

[21] D. Hall and T. Sands. (2020). Quantum Cryptography for Nuclear Command and Control. [Online]. Available: NavalPostgraduateSchool; GraduateSchoolofEngineeringandSciences(GSEAS)

[22] F. Xu, X. Ma, Q. Zhang, H.-K. Lo, and J.-W. Pan, "Secure quantum key distribution with realistic devices," in *Reviews of Modern Physics*, 2020, vol. 92, pp. 025 002–(1–60).

[23] W. Wootters and W. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, pp. 802–803, Oct. 1982.

[24] Excelitas SPCM-AQRH Family datasheet. [Online]. Available: https://www.excelitas.com/product/spcm-aqrh. Accessed 29 Jun 2021.

[25] Micro Photon Devices In-GaAs/InP datasheet. [Online]. Available: http://www.micro-photon-devices.com/Products/Photon-Counters/InGaAs-InP. Accessed 07 Aug 2021.

[26] Single Quantum SNSPD datasheet. [Online]. Available: https://singlequantum.com/products/single-quantum-eos/. Accessed 07 Aug 2021.

[27] G. von Zengen, K. Garlichs, Y. Schrcöder, and L. C. Wolf, "A sub-microsecond clock synchronization protocol for wireless industrial monitoring and control networks," in *2017 IEEE International Conference on Industrial Technology (ICIT)*, 2017, pp. 1266–1270.

[28] Qubitekk Quantum Mechanics Lab Kit. (2021). [Online]. Available: http://qubitekk.com/products/quantum-mechanics-lab-kit/. Accessed 29 Jun 2021.

[29] C. Hong, Z. Ou, and L. Mandel, "Measurement of subpicosecond time intervals between two photons by interference," *Physical Review Letters*, vol. 59, no. 18, pp. 2044–2046, Nov. 1987.

[30] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," in *Reviews of Modern Physics*, 2002, vol. 74, pp. 145–195.

[31] C. E. Shannon, "Communication theory of secrecy systems," *The Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, Oct. 1949.

[32] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transaction on Information Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.

[33] R. L. Rivest, A. Shamir, and L. M. Adleman, "Cryptographic communications system and method," U.S. Patent 4 405 829, Sep. 20, 1983.

[34] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," in *Annual Symposium on Foundations of Computer Science*, 1994, vol. 35, pp. 124–134.

[35] L. Valenta, S. Cohney, A. Liao, J. Fried, S. Bodduluri, and N. Heninger, "Factoring as a service," 05 2017, pp. 321–338.

[36] S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffett, C. Putnam, Craig, and P. Zimmermann, "Factorization of a 512-bit RSA modulus," in *Advances in Cryptology — EUROCRYPT 2000*, B. Preneel, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 1–18.

[37] T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. te Riele, A. Timofeev, and P. Zimmermann, "Factorization of a 768-bit RSA modulus," in *Advances in Cryptology – CRYPTO 2010*, T. Rabin, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 333–350.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California