Quantum Key Distribution Data Post-Processing with Limited Resources: Towards Satellite-Based Quantum Communication

by

Nikolay Gigov

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering - Quantum Information

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Nikolay Gigov

Abstract

Quantum key distribution (QKD), a novel cryptographic technique for secure distribution of secret keys between two parties, is the first successful quantum technology to emerge from quantum information science. The security of QKD is guaranteed by fundamental properties of quantum mechanical systems, unlike public-key cryptography whose security depends on difficult to solve mathematical problems such as factoring. Current terrestrial quantum links are limited to about 250 km. However, QKD could soon be deployed on a global scale over free-space links to an orbiting satellite used as a trusted node.

Envisioning a photonic uplink to a quantum receiver positioned on a low Earth orbit satellite, the Canadian Quantum Encryption and Science Satellite (QEYSSat) is a collaborative project involving Canadian universities, the Canadian Space Agency (CSA) and industry partners. This thesis presents some of the research conducted towards feasibility studies of the QEYSSat mission.

One of the main goals of this research is to develop technologies for data acquisition and processing required for a satellite-based QKD system. A working testbed system helps to establish firmly grounded estimates of the overall complexity, the computing resources necessary, and the bandwidth requirements of the classical communication channel. It can also serve as a good foundation for the design and development of a future payload computer onboard QEYSSat.

This thesis describes the design and implementation of a QKD post-processing system which aims to minimize the computing requirements at one side of the link, unlike most traditional implementations which assume symmetric computing resources at each end. The post-processing software features precise coincidence analysis, error correction based on low-density parity-check codes, privacy amplification employing Toeplitz hash functions, and a procedure for automated polarization alignment.

The system's hardware and software components integrate fully with a quantum optical apparatus used to demonstrate the feasibility of QKD with a satellite uplink. Detailed computing resource requirements and QKD results from the operation of the entire system at high-loss regimes are presented here.

Acknowledgements

I would like to thank Dr. Thomas Jennewein, Dr. Amir Khandani and Dr. Norbert Lütkenhaus for their guidance and support. I have enjoyed insightful discussions and collaborations with many members of Dr. Jennewein's research group. Specifically, I would like to thank Christopher Erven for his help with the QKD software, Brendon Higgins for implementing an automated polarization alignment library (which I incorporated in the QKD software), Jean-Philippe Bourgoin for his invaluable help with the high-loss QKD experiment and his satellite link analysis, Zhizhong Yan for his work on the source modulator, and Evan Meyer-Scott for answering my many questions.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

This thesis is organized into five main chapters. The current chapter provides background information on quantum key distribution (QKD) focusing on theoretical aspects of the protocols as well as some practical aspects of satellite-based QKD. Chapter 2 describes the experimental apparatus employed to model QKD with a satellite uplink. Chapter 3 details the design and implementation of a QKD post-processing system which aims to minimize the computing requirements at the satellite side of the link. Chapter 4 provides estimates of the computing resources required onboard the satellite as well as the classical communication requirements. Finally, in chapter 5, a detailed performance analysis of the entire QKD testbed system is presented.

1.1 Secure Communication

Secure communication is paramount to modern society. The rapid growth of electronic transactions and interactions has come with rising concerns about security and privacy. All kinds of information, from personal data to trade secrets, is regularly transmitted over public communication channels, and yet, it needs to be protected from unauthorized use. Secure communication is by no means new to the modern age, however, lately, its importance has greatly increased due to the sheer amount of sensitive data exchanged on a daily basis.

For centuries, many different cryptographic techniques have been employed (with variable success) to ensure confidentiality between communicating parties. Today, we rely heavily on public-key cryptosystems and protocols such as RSA, TLS, AES, to name a few. Although very practical and suitable for high traffic rates, most of those protocols do not actually guarantee provable security. Instead, they are deemed secure, because to

break them an adversary needs to solve difficult mathematical problems such as factoring of large integers. The measure of security in those schemes is the amount of computational power available to the adversary and the time necessary to solve the underlying problems [1]. Thus, cryptographic standards need to be regularly updated as computing technology moves forward.

There is one classical encryption scheme which is provably secure. Invented in 1882 by Frank Miller, re-discovered and patented in 1917 by Gilbert Vernam, and proven optimal in 1949 by Shannon [2], the Vernam cipher is a symmetric-key scheme in which both communicating parties share a random secret key (or pad). To encrypt messages, the key bits are simply XOR-ed[1] with the message bits. Security is only guaranteed if the secret key is used once, and hence the scheme is also known as the *one-time pad*. Thus, the size of the shared key has to be greater than or equal to the length of the message—a major limitation which has prevented the widespread use of one-time pad encryption. In other words, for this scheme to be employed successfully, large amounts of key must be securely distributed among communicating parties [1]. Classically, there is no reliable way to do so other than physically transporting the key in a safe way (e.g. via trusted couriers), however, quantum information science has turned up a surprising solution which is discussed in the next section.

1.2 Quantum Key Distribution

Quantum key distribution (QKD), a novel cryptographic technique for secure distribution of secret keys between two parties (traditionally named Alice and Bob), is the first successful quantum technology to emerge from quantum information science. QKD employs quantum states to encode and transmit secure key bits. The security of QKD is guaranteed by fundamental properties of quantum mechanical systems. Intuitively, any intermediate measurement of a quantum state disturbs that state.

More formally, the *no-cloning*[2] property [3] of quantum states guarantees that an adversary cannot obtain a copy of the secret key without introducing detectable errors in the system. The amount of errors in the quantum communication channel is described by the quantum bit error ratio (QBER) which is analogous to the BER of classical channels. If the QBER is above a certain threshold (near 17% [1]), Alice and Bob assume that

[1] The bitwise XOR operation, often denoted with a $\oplus$ symbol, is equivalent to bitwise addition modulo two.

[2] The No-Cloning theorem [3] of quantum information asserts that, assuming the rules of quantum mechanics, it is not possible to copy an unknown quantum state with perfect fidelity.

the eavesdropper (commonly named Eve) has intercepted the communication, and hence the quantum exchange must be restarted. For QBER values below the threshold, the legitimate parties apply classical post-processing techniques to clean the key of errors (see section 3.6) and to reduce Eve's information about the key below an infinitesimal value (see section 3.7). For a comprehensive review of QKD and its security, see for example [1].

Unlike classical cryptography, QKD is able to provide long-term security [4] in addition to being immune against future attacks employing quantum computers[3]. If an eavesdropper does not acquire the secret key during its establishment, there is no amount of computational power which will allow her to obtain it at a later time. This statement is not true for common cryptographic protocols. If Eve were to capture RSA-encrypted [6] data and corresponding public-key communication today, she could start solving the underlying factoring problem and, within a few years, gain access to the secret message which may still contain valid sensitive information.

1.2.1 QKD Protocols

As mentioned, QKD utilizes quantum states as carriers of secure key bits. In principle, the state of any quantum mechanical system could be employed. In practice, quantum states of light are almost always used [1], because they can be transmitted at the fastest possible speed with little decoherence over an optical fiber [7] or a free-space link [8] with direct line of sight between the transmitter and the receiver. Most QKD protocols are *discrete-variable*, based on photon polarization states, however, *continuous-variable* [9, 10] ones, in which real amplitudes are measured instead of discrete events [1], have also been developed and demonstrated. In general, QKD protocols accomplish the following three steps:

1. *Raw Key Exchange*: Quantum states are created, transmitted over a quantum channel, and measured to establish a *raw key*.

2. *Key Sifting*: Only a subset of the measurements are selected according to the specifics of the protocol. This step produces the *sifted key*.

3. *Classical Post-Processing*: Also known as key distillation, this step consists of classical procedures called error correction and privacy amplification, which are employed to clean the sifted key of errors and to eliminate the information leaked out to the

[3] In 1994, Peter Shor formulated a quantum algorithm for fast (polynomial-time) factorization of integers[5]. If and when scalable quantum computers are built, many commercial cryptographic schemes such as RSA will be rendered completely insecure.

Polarization State	Dirac Notation	Basis	Bit Value
Horizontal	$\|H\rangle$	H/V (+)	0
Vertical	$\|V\rangle$	H/V (+)	1
Diagonal	$\|D\rangle$	D/A (×)	0
Anti-Diagonal	$\|A\rangle$	D/A (×)	1

Table 1.1: BB84 polarization state encoding.

adversary (see chapter 3). At the end of this step, Alice and Bob share an identical *secure key*, or, if the QBER is above a security threshold determined by formal proofs of the QKD protocol's security [1], the sifted key is discarded and the protocol restarts at step 1. Note that Alice and Bob require an *authenticated*[4] classical communication channel to complete this step.

QKD protocols differ only in the first two steps, while classical post-processing is generally studied separately.

BB84

The first QKD protocol, BB84, is named after its creators Bennett and Brassard who invented QKD in 1984. BB84 is a *prepare-and-measure* scheme in which Alice encodes each bit from her random key in one of the four polarization states as displayed in table 1.1. Given this encoding, the protocol consists of the following steps [1]:

1. *Raw Key Exchange*: Alice uses a random number generator to create a random key. Employing a single-photon source, she encodes each bit of the key (0 or 1) with the polarization state encoding from table 1.1 and transmits the photon over to Bob via the quantum channel. She makes random basis choices which she records. Bob measures each received photon randomly in either the + or × basis.

2. *Key Sifting*: Alice and Bob communicate their basis choices over a classical channel. They discard those measurements (about half) which are performed in non-matching bases.

3. *Classical Post-Processing*: As in the general case, Alice and Bob perform error correction (EC) and privacy amplification (PA) using the classical communication channel.

[4] Authentication prevents Eve from modifying any messages exchanged over the classical communication channel. A small amount of secret key is needed to authenticate each party. That is why QKD is sometimes referred to as a *key-growing* procedure.

If the QBER is above the security threshold, the sifted key is discarded and the protocol is restarted.

1.2.2 BB84 with Decoy States

A significant drawback of BB84 as formulated above is the assumption that each information carrier is a single photon. In practice, it is very difficult to create high-rate sources of perfect single photons.

In quantum mechanics, states containing exactly n photons are known as Fock states, denoted as $|n\rangle$ in Dirac notation. Pulsed lasers are the most convenient high-frequency and high-intensity sources of monochromatic light for QKD. However, instead of Fock states, a pulsed laser generates *coherent* states of the form

$$|\sqrt{\mu}\, e^{i\theta}\rangle \equiv |\alpha\rangle = e^{-\mu/2} \sum_{n=0}^{\infty} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \tag{1.1}$$

where $\mu = |\alpha|^2$ is the average photon number [1] per laser pulse. Thus, there is a non-zero probability that each resulting pulse of light contains more than one photon. This imperfection gives rise to the photon number splitting attack [11], in which Eve captures one photon from each multi-photon pulse, stores it, and measures it (in the correct basis) after Bob's basis choices have been revealed. Hence, to minimize the multi-photon emission probability, the laser pulses are attenuated and phase-randomized. This kind of source, called a weak coherent pulse (WCP) source, is Poissonian, and the probability that a pulse contains n photons is given by

$$P(n|\mu) = \frac{\mu^n}{n!}\, e^{-\mu} \tag{1.2}$$

However, signal attenuation severely limits the length of the quantum channel and hence the total distance over which QKD can be performed. Decoy states [12, 13] were introduced to overcome this practical problem. The main idea is that Alice can randomly vary the average photon number of her coherent pulses, while the polarization encoding remains unchanged from table 1.1. There are different decoy-state schemes, but in the simplest case, there are only two kinds of states: *signal* states with average photon number μ, and *decoy* states with average photon number ν. Only signal states contribute towards secure key generation. Both μ and ν still need to be smaller than one to keep the multi-photon probability low [14], but less attenuation is required for signal pulses (e.g. $\mu = 0.5$) compared to BB84 without decoy states [13].

Since the adversary has no way of measuring the average photon number levels, this scheme allows the legitimate parties to bound Eve's information obtained from multi-photon events. The bound is provided by information-theoretic security proofs [14] and ultimately relates to the required amount of privacy amplification. A detailed example of one such protocol and its theoretic bounds is discussed below.

Vacuum+Weak Decoy-State Protocol

The QKD experiment described in this thesis (see chapter 2) implements the vacuum+weak decoy-state protocol [14], in which the source randomly emits signal states with average photon number μ, or decoy states that are either vacuum "pulses" or have an average photon number $\nu < \mu$. The authors of [14] show that this protocol behaves close to optimal in their simulations.

In our implementation, Alice employs a polarization and intensity modulator [15] (see section 2.2.1), to prepare a random sequence of BB84 polarization encodings which are 92% signal and 8% decoy states; vacuum states are "sent" between successive laser pulses [16]. The average photon number values ($\mu = 0.5$ and $\nu = 0.1$) are chosen according to the optimization procedure described in [14].

According to the asymptotic-key formalism from [14], the lower bound for the final asymptotic secure key rate per laser pulse, R is given by

$$R = qL_{\mu\nu}\left\{-Q_\mu\, \eta_{\mathrm{EC}}\left(E_\mu\right) H_2\left(E_\mu\right) + Q_1^L\left[1 - H_2\left(E_1^U\right)\right]\right\} \tag{1.3}$$

where

- q is a basis reconciliation factor, which for BB84 is equal to 1/2, since we expect Alice and Bob to agree on their basis choices for only half of the measurements.
- $L_{\mu\nu} = \frac{N_\mu}{N_\mu + N_\nu}$ is the fraction of all photon detections attributed to signal states.
- $Q_{\mu/\nu}$ is the gain for signal/decoy states. The gain is calculated as the ratio of number of photons received by Bob to number of pulses sent by Alice.
- $\eta_{\mathrm{EC}}\left(E_\mu\right)$ is the efficiency parameter of the error correction algorithm (see section 3.6).
- H_2 is the binary entropy function. E_μ is the QBER estimate for signal states.
- Q_1^L and E_1^L are lower bounds of the gain and QBER for single photon pulses.

The lower bound of the single-photon gain, Q_1^L is calculated [14] as

$$Q_1^L = \frac{\mu^2\, e^{-\mu}}{\mu\nu - \nu^2}\left(Q_\nu\, e^\nu - Q_\mu\, e^\mu \frac{\nu^2}{\mu^2} - \frac{\mu^2 - \nu^2}{\mu^2} Y_0\right) \tag{1.4}$$

where Y_0 is the vacuum yield. Y_0 is determined by the cumulative rate of detector dark counts and background noise within the coincidence window (see section 3.4), and is measured between successive laser pulses.

The upper bound on the QBER, E_1^{U} for single photon states is calculated [14, 17] as

$$E_1^{\mathrm{U}}(\mu) = \frac{E_\nu Q_\nu \mathrm{e}^\nu - E_0 Y_0}{\nu Q_1^L} \mu\, e^{-\mu} \tag{1.5}$$

$$E_1^{\mathrm{U}}(\nu) = \frac{E_\nu Q_\nu}{Q_1^L} - \frac{E_0 Y_0}{Q_1^L\, e^\mu} \tag{1.6}$$

$$E_1^{\mathrm{U}} = \min\left\{E_1^{\mathrm{U}}(\mu),\ E_1^{\mathrm{U}}(\nu)\right\} \tag{1.7}$$

where E_ν is the QBER estimate for decoy states, and $E_0 = 0.5$ is the vacuum error rate.

In our experiment, the parameters in equations (1.4) and (1.7) are estimated from experimental data to obtain the asymptotic lower bound for the secret key rate per laser pulse, R, as defined in equation (1.3). To obtain the secure key rate in bits per secoind, R is multiplied by the output rate of the WCP source (see section 2.2).

Finite-Size Effects

The expression for the secret key rate per laser pulse given in equation (1.3) follows the asymptotic-key formalist, in which the rate is calculated assuming the raw key is of infinite length. This assumption is obviously unrealistic for QKD implementations with finite resources. In practice, the raw key is broken up into blocks of certain size and the full protocol is performed on each block of key (see chapter 3). With terrestrial links, Alice and Bob can simply perform the quantum state exchange step until a sufficient amount of raw key is established in order to form a large-enough block. With satellite links however, the parties have only a limited time to establish a key, and significant losses in the channel lead to short raw keys over a single satellite passage [18] (see section 5.4.2).

The topic of QKD security proofs with finite resources has only been studied recently [19], and ongoing research continues to re-examine all protocols. Unfortunately,

decoy-state protocols are one of those QKD schemes for which the finite-key analysis is incomplete. Nevertheless, there have been several attempts at an approximate analysis [14, 17, 20, 21].

Based on finite-size analysis for single-photon sources, current approximations [17, 20, 21] arrive at the following correction to equation (1.3)

$$R = qL_{\mu\nu}\left\{-Q_\mu\,\eta_{\mathrm{EC}}\left(E_\mu\right)H_2\left(E_\mu\right) + Q_1^L\left[1 - H_2\left(E_1^U\right)\right] - Q_\mu\,\Delta(n)\right\} \tag{1.8}$$

where n is the number of bits in the raw key. $\Delta(n)$ is obtained as follows:

$$\Delta_1(n) = 7\sqrt{\frac{\log_2(2/\bar{\varepsilon})}{n}} + \frac{2}{n}\log_2(1/\varepsilon_{\mathrm{PA}}) \tag{1.9}$$

$$\Delta_2(n) = \frac{1}{n}\log_2(2/\varepsilon_{\mathrm{EC}}) \tag{1.10}$$

$$\Delta(n) = \Delta_1(n) + \Delta_2(n) \tag{1.11}$$

- $\Delta_1(n)$ is a correction accounting for the smooth min-entropy of n bits [19, 20], which is the first term in equation (1.9). The second term accounts for privacy amplification.
- $\Delta_2(n)$ is a correction accounting for additional information leaked out during error correction.
- The security parameters $\varepsilon_{\mathrm{EC}}$ and $\varepsilon_{\mathrm{PA}}$ are the failure probabilities of error correction and privacy amplification respectively. The *smoothing* parameter, $\bar{\varepsilon}$ comes from the theory of smooth min-entropy and is optimized numerically [17] along with $\varepsilon_{\mathrm{PE}}$.

In Cai and Scarani's formalism [17], the maximum failure probability of the entire QKD protocol is quantified by ε, and is equal to the sum of the failure probabilities for each step of the protocol:

$$\varepsilon = \varepsilon_{\mathrm{EC}} + \bar{\varepsilon} + n_{\mathrm{PE}}\,\varepsilon_{\mathrm{PE}} + \varepsilon_{\mathrm{PA}} \tag{1.12}$$

where n_{PE} is the total number of parameters which need to be estimated, and $\varepsilon_{\mathrm{PE}}$ is the failure probability of the parameter estimation procedure. In a sense, ε is a global security parameter (for the whole protocol) which can be set according to the cryptographer's failure tolerance requirements (e.g. $\varepsilon = 10^{-6}$).

Furthermore, in a real-world QKD implementation, all measured quantities will be fluctuating to some extent, so the theoretical bounds might be incorrect if averaged values are used in the formulas. In other words, another correction is necessary to account for *statistical fluctuations* of the parameters involved in the calculation of Eve's information. Several authors [18, 21–23] resort to a simple solution, in which 10 standard deviations are either added to or subtracted from each measured parameter so that the lowest possible bound for the secure key rate is obtained.

One major problem with this argument is that the amount of fluctuation allowed by the model has no direct relation to the security of the final key. Sun et al. [21] simply claim that $10\,\sigma$ should be "good enough" because the probability that the parameters fall outside the range (measured value $\pm 10\,\sigma$) is less than 10^{-25}. Other authors [22, 23] also tend to repeat the $10\,\sigma$ assumption with no rigorous justification.

Here is a simple example relating to our experiment (see chapter 2) which demonstrates that this approach might not be ideal. Our quantum receiver exhibits, on average, about 95 background events+dark counts per second. These counts follow a Poissonian distribution, so the standard deviation is $\sigma = \sqrt{95} \approx 9.7$, and $10\,\sigma \approx 97$. Hence, Y_0 is either reduced to 0 or doubled if 10 standard deviations are subtracted or added respectively. The problem is that, at very high channel loss (e.g. above 55 dB), our legitimate detection rates do not go far above the background rates, and thus doubling Y_0 essentially "cancels out" the signal.

A better approach, based on the law of large numbers, is proposed by Cai and Scarani [17]. Suppose a parameter $\lambda \in \{Q_{\mu/\nu},\ E_{\mu/\nu},\ Y_0\}$ is estimated with a finite number m of samples. Then the estimate λ_m differs from the ideal value λ_∞ by at most $\delta(m, d)$:

$$|\lambda_m - \lambda_\infty| \leq \delta(m,d) \equiv \frac{1}{2}\sqrt{\frac{2\,\ln(1/\varepsilon_{\mathrm{PE}}) + d\,\ln(m+1)}{m}} \tag{1.13}$$

where d is the number of outcomes from the measurement[5] of λ_m. The upper and lower bounds for each parameter are then given by

$$\lambda^U = \min(\lambda + \delta, 1), \quad \lambda^L = \max(\lambda - \delta, 0) \tag{1.14}$$

As mentioned, this analysis still provides only an estimate of the finite-size effects on the secret key rate. Note that each effect is treated separately and statistical fluctuations are added-on at the end. So far, for decoy-state QKD, there has been no rigorous treatment which considers statistical fluctuations from the onset of the security proof.

[5] A generalized measurement, which in quantum mechanics is performed with a positive-operator valued measure (POVM) [17]. For example, $d = 2$ for qubit measurements.

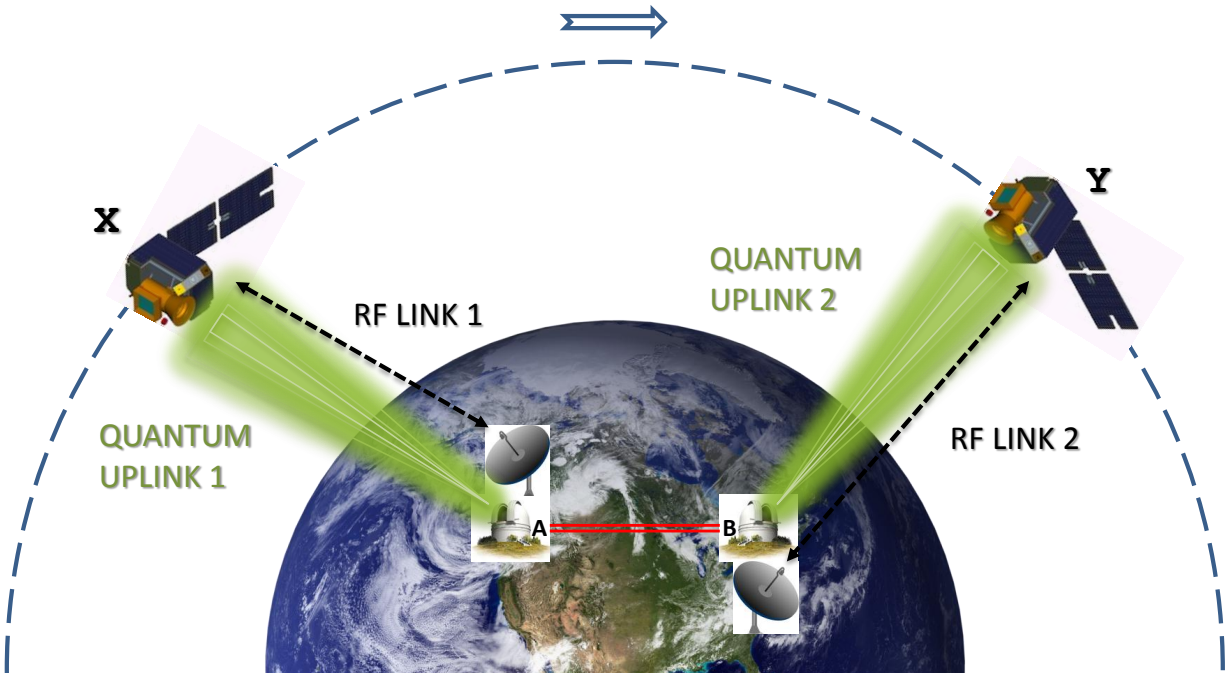

Figure 1.1: Illustration of QKD with a satellite used as a trusted node. An orbiting LEO satellite allows locations A and B to communicate securely over a classical link (red) by establishing a secret key with each location through a free-space QKD uplink.

1.3 Satellite-Based QKD

1.3.1 Motivation and Overview

Although QKD has been demonstrated experimentally and commercially deployed [24, 25], current implementations are range-limited to quantum links of up to 260 km in optical fiber [7, 26, 27] and 144 km over free-space [28]. Despite future technological advances, the range of point-to-point terrestrial links is not expected to grow past 400 km [27]. Beyond this distance, the signal-to-noise ratio of the quantum channel drops below practical values due to transmission losses and decoherence effects. In fiber links for example, the loss grows exponentially with distance [14]. Free-space links are limited by the the curvature of the Earth, as well as by atmospheric transmission losses.

In classical communications systems, optical repeaters/amplifiers are commonly employed to periodically boost the classical signal along a long-distance link. Due to the no-cloning theorem, they cannot be used for QKD. Instead, specially designed *quantum* repeaters [29] are required for quantum signals. However, such repeaters require quantum

memories in order to implement the entanglement swapping protocol—a procedure for transferring the quantum properties of one quantum system (e.g. photon) onto another via intermediate quantum teleportation steps. Quantum memories are unfortunately still at the fundamental research stage and nowhere near the technological maturity [30] necessary for practical QKD applications.

An alternative solution to this scalability problem is to deploy QKD over free-space links to satellite platforms. Such orbiting intermediate nodes can then bridge multiple local QKD networks [31, 32] on the ground in order to form a global QKD network. In recent years, there have been several different proposals [28, 33–41] for satellite-based QKD involving one or more spacecraft, in either geostationary Earth orbit (GEO) or low Earth orbit (LEO), used as either trusted or untrusted nodes.

Even though multi-satellite proposals exist[42], initial proof-of-principle missions will likely employ just one. One intuitive approach is to put a source of entangled photons on an orbiting spacecraft and use a protocol such as BBM92 [43]. Conversely, if we preserve the same triangular configuration and reverse the links, quantum uplinks from the ground could be combined in a Bell state measurement [44] on the satellite, potentially demonstrating *device-independent* QKD [45]. The advantage of these schemes is that the satellite does not need to be trusted—an eavesdropper can be detected even if she has full control of the satellite. The downside is that, to maximize the total key distribution distance and elevation angles [18], the spacecraft needs to be placed far away from Earth, likely in GEO. Such distances and associated losses are unfortunately not yet possible to overcome technologically.

Alternatively, we can loosen the security requirement a bit and treat the satellite as a trusted node. In that case, an LEO (altitude of up to 1000 km) spacecraft can establish a secret key (e.g. via the BB84 protocol) with one or more ground stations as it orbits around the Earth. In the simplest case, two distant locations A and B can establish a secret key, $\mathbf{X}$ and $\mathbf{Y}$ respectively, with the satellite as illustrated in figure 1.1. For A and B to share a key, the satellite needs to send $(\mathbf{X} \oplus \mathbf{Y})$ to B (over the classical channel), so that B can obtain $\mathbf{X}$ by using its key $\mathbf{Y}$ as follows:

$$(\mathbf{X} \oplus \mathbf{Y}) \oplus \mathbf{Y} = \mathbf{X} \oplus (\mathbf{Y} \oplus \mathbf{Y}) \tag{1.15}$$

$$= \mathbf{X} \oplus \mathbf{0} \tag{1.16}$$

$$= \mathbf{X} \tag{1.17}$$

The above procedure essentially amounts to one-time pad encryption (the satellite encrypts A's key with B's key) and decryption (see section 1.1).

This trusted-node scheme is the most technologically feasible in the near term due to the shorter free-space link compared to GEO (leading to tolerable channel loss) as well as the overall reduced complexity (one link established at a time). Its main drawback is that it is less secure—it requires that the satellite store the secret key obtained from location A until the key with ground station B is established. Hence, extra precautions must be taken to ensure that an eavesdropper cannot access the (classical) memory on the spacecraft.

1.3.2 The QEYSSat Mission Proposal

A number of satellite-based quantum communications missions [46–49] are currently under development and expected to launch in the near future [47]. One such mission is the Canadian Quantum Encryption and Science Satellite (QEYSSat) which envisions a photonic uplink to an LEO microsatellite that acts as a trusted node for QKD and a platform for fundamental quantum physics experiments [49, 50]. QEYSSat is a collaborative project involving Canadian universities, the Canadian Space Agency (CSA) and several industry partners (COM DEV, INO, Neptec).

The proposed spacecraft will carry a special "quantum payload" comprising a quantum receiver (similar to the one described in section 2.3) capable of analyzing photon polarization, as well as an integrated QKD data acquisition, processing and key management system necessary to implement the trusted-node scheme illustrated in figure 1.1. The satellite will receive photonic signals sent upwards from optical ground stations, while dedicated RF stations nearby will provide two-way classical communication links. In the current proposal, QEYSSat is based on COM DEV's Advanced Integrated Microsatellite (AIM) bus employed on a similar spacecraft, the Maritime Monitoring and Messaging Microsatellite (M3MSat) [50].

An important consideration for this mission is the choice between a photonic quantum uplink (source located on the ground) or downlink (source located on the satellite). Detailed analysis and simulations of each link scenario can be found in a recent publication by Bourgoin et al. [18]. Most studies in the past have concentrated solely on downlink proposals, based on the fact that uplinks suffer from additional loss induced by atmospheric turbulence. The effects of turbulence on beam propagation are mostly negligible above the first 20 km of Earth's atmosphere [51], and hence, uplinks and downlink are affected differently. In a downlink, turbulence is present only at the end of the transmission path and has little impact on the light beam. For an uplink however, turbulence can be a significant problem, which can be resolved by adaptive optics and/or a good choice of ground station location [18]. Further details and analysis of all factors affecting beam propagation and channel loss can be found in [18].

On the other hand, an uplink implementation is very appealing from a practical point of view. Current high-frequency photon sources are very complex, have relatively high power requirements, and often require regular maintenance; all these factors make them unsuitable (see appendix A) as a satellite payload. Conversely, a source on the ground can be easily updated as technology progresses. This versatile design allows for different types of sources to be used, and a range of quantum mechanics experiments to be performed [52]. Furthermore, a quantum receiver is much simpler than a source and many of its components, such as single-photon detectors, have already flown in space [53]. The uplink scenario has many additional advantages owing to its reduced complexity and lower storage, processing, and classical communications requirements (see chapter 4).

In the following chapters, this thesis presents some of the research conducted towards feasibility studies of the QEYSSat mission. In particular, it focuses on the necessary data processing algorithms (chapter 3) and their computing resource requirements (chapter 4) on the satellite. It also details the design (chapter 2) and performance (chapter 5) of a quantum optical apparatus used to demonstrate the feasibility of QKD with a satellite uplink.

Chapter 2

Experimental Apparatus Demonstrating Satellite Uplink Feasibility

This chapter describes the experimental setup employed to model QKD with a satellite uplink. Our experiment builds upon an earlier version of the apparatus, detailed in [15, 16, 54]. Its recent evolution includes the addition of a full quantum optical receiver, automated polarization alignment, more realistic timing synchronization and the implementation of a full QKD protocol.

Author contributions

Evan Meyer-Scott built and characterized the original photon source [16, 54]. Zhizhong Yan developed the FPGA-controlled telecom intensity and polarization modulator [15]. Jean-Philippe Bourgoin and Brendon Higgins designed and constructed the new quantum receiver. Thomas Jennewein and Brendon Higgins provided guidance and supervision throughout the project.

My main contribution to this experiment was in efficient post-processing as described in detail in chapter 3. I added more hardware for timing analysis and integrated the live data streams coming from the source, receiver, time-taggers and GPS units into the QKD software. I worked with Zhizhong Yan to analyze the polarization state distribution and QBER stability of the telecom modulator (section 2.2.1), and I optimized the decoy-state levels for better key rates. I performed the QKD experiment together with Jean-Philippe Bourgoin, and analyzed the data using my newly developed QKD post-processing system (see chapter 3).

2.1 Overview

The *quantum* components of a practical QKD system typically consist of a quantum source, a transmission medium (quantum channel) and a quantum receiver. Most discrete-variable QKD implementations employ either a weak coherent pulse (WCP) source [7, 55–58], or a source emitting entangled photon pairs [42, 59–63]. The quantum signals are transmitted either over optical fiber [7] or over a free-space link [8] with direct line of sight between the transmitter and the receiver. Quantum receivers typically incorporate two or more photon detectors which are characterized by their detection efficiency (at a given wavelength) and dark count rate[1].

In the challenging case of a satellite uplink (see section 1.3), the quantum source is required to have a very high repetition rate to overcome the considerable losses in the free-space channel. In addition, the emitted photons should have short pulse widths to enable accurate temporal filtering (see section 3.4). A decoy-state WCP source implemented with a mode-locked laser, such as a titanium-sapphire (Ti:Sapph) laser, can satisfy these requirements. However, this kind of source produces unwanted phase correlations between consecutive laser pulses, and thus violates an important assumption of QKD security proofs [57, 64].

To overcome this shortcoming, our source has a hybrid design based on a process known as sum-frequency generation (SFG) or *upconversion* in which the absorption of multiple photons of lower energy (longer wavelength) results in the emission of higher-energy (shorter wavelength) photons. As described in [15, 16, 54] and section 2.2.2, the source combines two laser beams—one coming from a mode-locked Ti:Sapph laser at 810 nm, and another generated by a continuous wave (CW) telecom laser operating at 1550 nm—to produce photons at 532 nm. In this design, phase randomization is provided by the CW telecom laser. Intensity and polarization modulation (section 2.2.1) is also accomplished at the telecom band for which fast and reliable commercial modulation components are readily available. The resulting wavelength (532 nm) of the output light is engineered in such a way as to be suitable for single photon detectors with the best figure of merit [65].

The output photons are sent over a short ($\approx$ 1 m) free-space channel with controllable loss, which attempts to emulate the high-loss environment of a satellite uplink. After the lossy link, the remaining photons are detected with our quantum receiver (section 2.3). The receiver contains low-dark-count thin silicon avalanche photo diode (Si-APD) detectors whose efficiency is optimal for photons of wavelength around 532 nm.

[1] A detector dark count is a false-positive detection event which is intrinsic to the detector and not caused by external light.

2.2 Weak Coherent Pulse Source

Our weak coherent pulse (WCP) source consists of two subsystems, an FPGA-controlled intensity and polarization telecom modulator (section 2.2.1) and an upconversion component (section 2.2.2). Both components work in unison to produce a pseudo-random sequence of polarization states encoded in short, phase-correlation-free light pulses.

2.2.1 Modulation Subsystem

Our high-speed telecom modulation system is built with commercially available ectro-optic (EO) amplitude and phase, lithium niobate waveguide modulators arranged in a balanced Mach-Zehnder interferometer (MZI) configuration. As depicted in figure 2.1 (a), the polarization of the input light coming from a tunable CW telecom laser is first manually adjusted (via FPC_1) so that each arm of the polarization beam splitter (PBS) receives equal beam intensity. Before entering the insulated and temperature-controlled box, the beam goes through the intensity modulator which implements signal and decoy states (see section 1.2.2). Within the box, each MZI arm contains a phase modulator. This configuration allows, for example, the control of the relative phase between H and V polarizations produced when the two beams recombine at the polarization beam combiner (PBC). Thus, any of the four polarization states (H, V, D or A) required for decoy-state BB84 can be generated by applying a certain combination of voltage settings to the EO phase modulators, while the signal/decoy levels are controlled by the voltage setting of the intensity modulator.

The settings of all modulators are controlled by a field programmable gate array (FPGA) driver circuit. The FPGA system clock is driven by an external 76 MHz TTL signal derived from the mode-locked Ti:Sapph laser and a pulse shaping circuit. The digital encodings produced by the FPGA are translated to corresponding analog signals via fast digital-to-analog converter (D/A) circuits. Finally, the analog signals are amplified by three RF driving amplifiers (DRVs) which are DC coupled to the corresponding RF ports (IM, PM_A and PM_B) on the EO modulators.

The FPGA is programmed to produce a pseudo-random sequence of polarization states (H, V, D or A) in which the signal ($\mu = 0.5$) or decoy ($\nu = 0.1$) level settings of the average photon number per pulse also vary in a pseudo-random way. Furthermore, decoy states account for about 8% of the total number of modulated pulses. For timing synchronization, the FPGA board periodically produces a single TTL signal indicating that 128 states have been modulated. This signal is accurately timed-stamped by Alice's time-tagging unit and

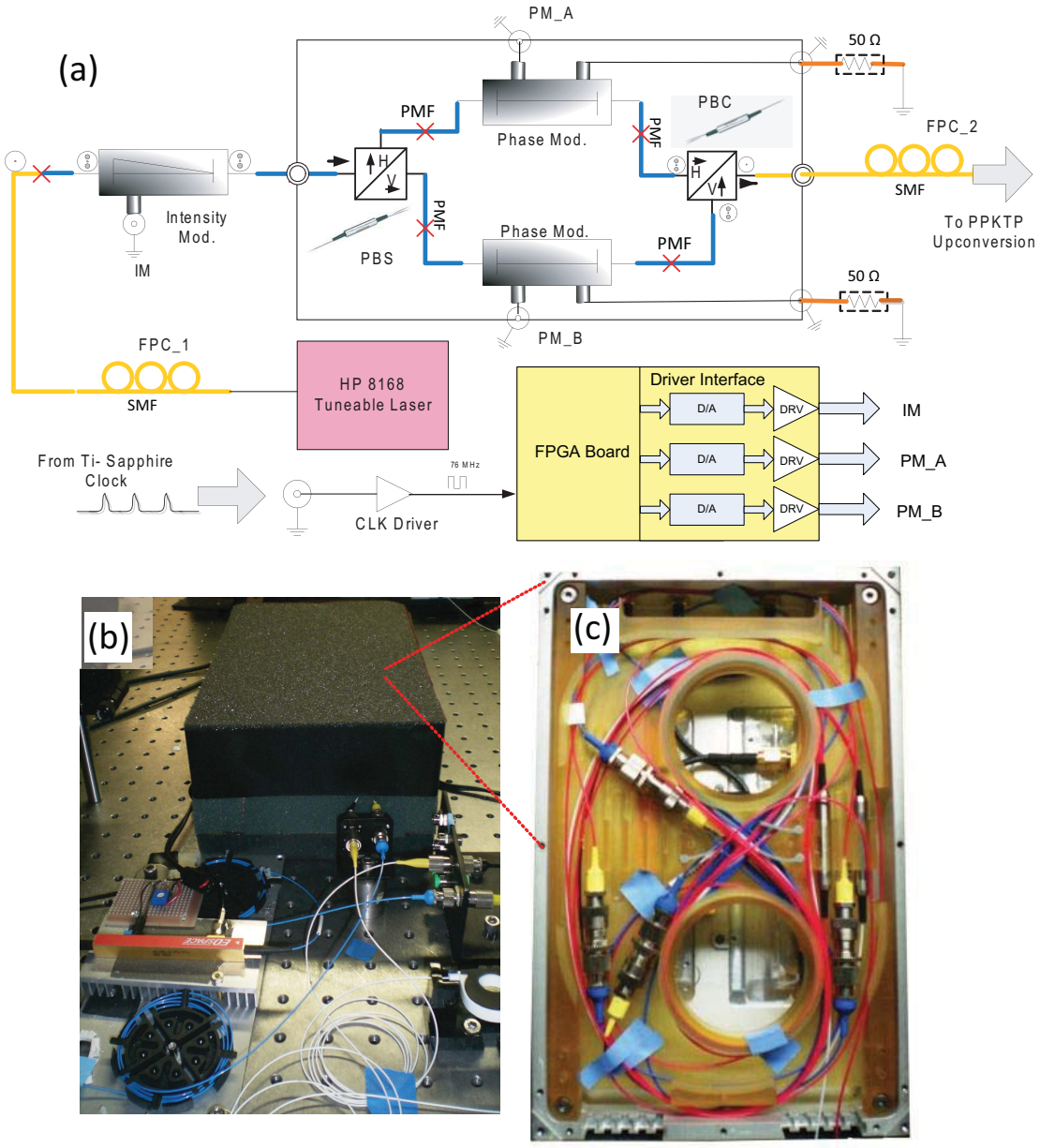

Figure 2.1: (a) Schematic representation of the FPGA-controlled intensity and polarization telecom modulator from [15]. Yellow and blue lines represent standard single-mode and polarization-maintaining fiber (SMF and PMF) respectively. Red × marks denote FC/PC connectors. Manual fiber polarization controllers (FPC_1 and FPC_2) are located at the input and output fibers. A mode-locked Ti:Sapph laser with repetition rate of ≈76 MHz drives the FPGA controller board, which in turn drives the intensity (IM), and phase (PM_A and PM_B) modulators. (b) and (c) Photographs of the modulators and polarization beam splitters/combiners (PBS/PBC) where (c) shows the interior of the insulated box, temperature-controlled by a Thorlabs TC-200 controller, containing two phase modulators, a PBS and a PBC.

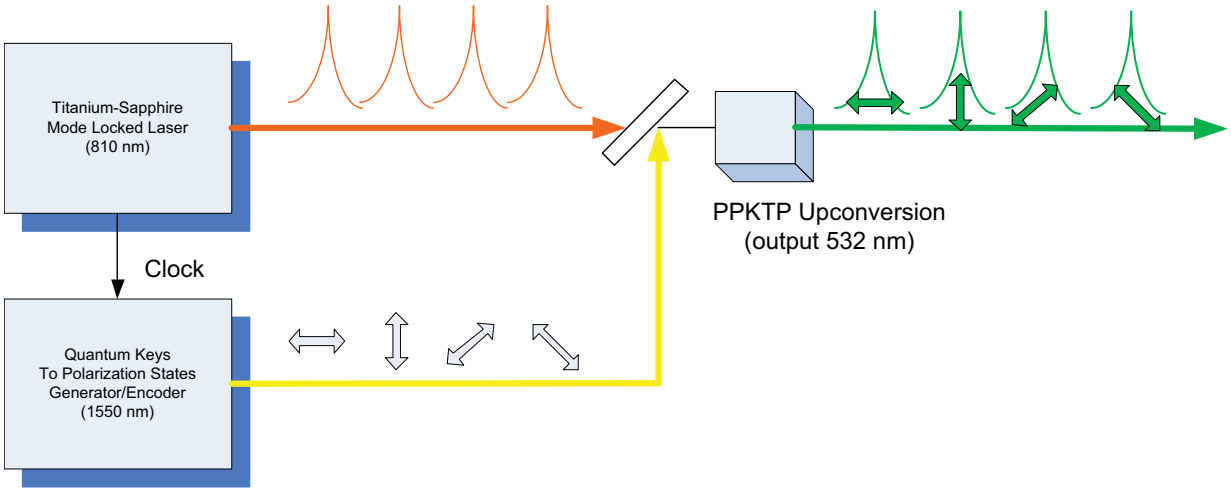

Figure 2.2: Generation of decoy-state BB84 states through upconversion of two pump laser beams in a polarization-compensated PPKTP crystal pair. The 810 nm beam (marked red) is produced by a pulsed Ti:Sapph laser, while the modulated 1550 nm beam (marked yellow) originates from a CW telecom laser. The resulting photons (green) at 532 nm inherit desirable properties from each pump source: the Ti:Sapph laser provides high repetition rate and short pulse width, while phase randomization, and intensity and polarization modulation are accomplished on the telecom side. Figure from [15].

is eventually used by the coincidence algorithm (see section 3.4) to generate matching pairs of photon emission–detection events.

2.2.2 Upconversion Subsystem

Once the 1550 nm photon beam has been modulated (as per the previous section), it is combined through SFG with another beam coming from a pulsed Ti:Sapph laser at 810 nm to produce upconverted green photons at 532 nm. The two laser beams are brought together into two orthogonal type-I periodically-poled (PP) potassium titanyl phosphate (KTP) crystals. The resulting upconversion process is illustrated in figure 2.2. Details on how SFG works can be found, for example, in [54]. In accordance with our decoy-state BB84 QKD scheme (section 1.2.2), the beam power of each pump laser is adjusted so that the average photon number of the output pulses is around 0.47.

The main idea behind this hybrid design is to impart desirable properties from each pump source to the resulting photons. As mentioned, the Ti:Sapph laser provides high repetition rate and short pulse width, while the CW telecom laser ensures phase randomization. The polarization of the 810 nm photons coming from the Ti:Sapph is fixed at $+45°$, i.e. at the diagonal (D) state:

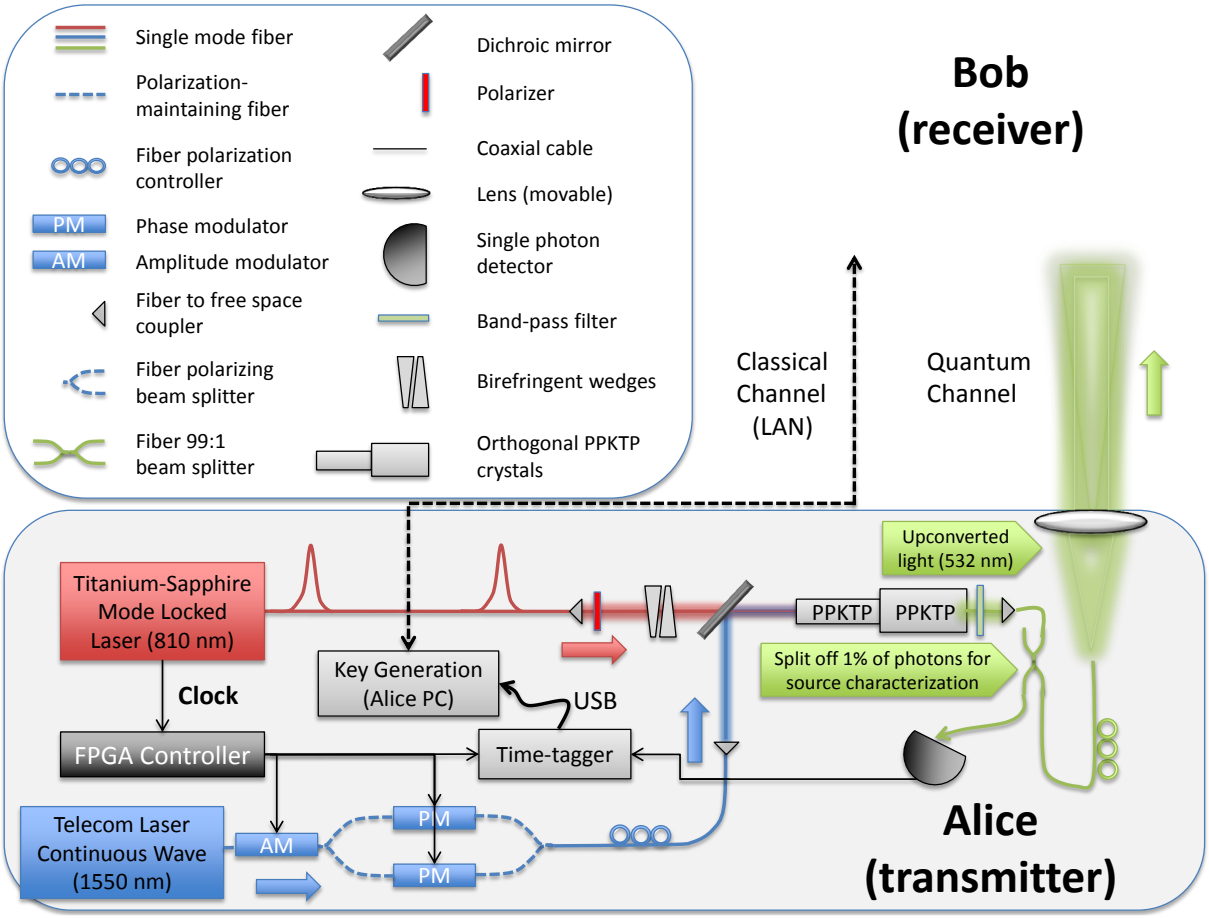

Figure 2.3: System-level schematic of the WCP source as part of the overall decoy-state BB84 QKD system. After Alice produces her desired photon states via telecom modulation and upconversion, she measures 1% of the green photons for source characterization, discards 9% and sends the remaining 90% of upconverted photons over the quantum channel where the loss is controlled through a movable lens. Bob then detects transmitted light with the quantum receiver. Both parties communicate classically over the LAN. Figure modified from [16, 54] with permission.

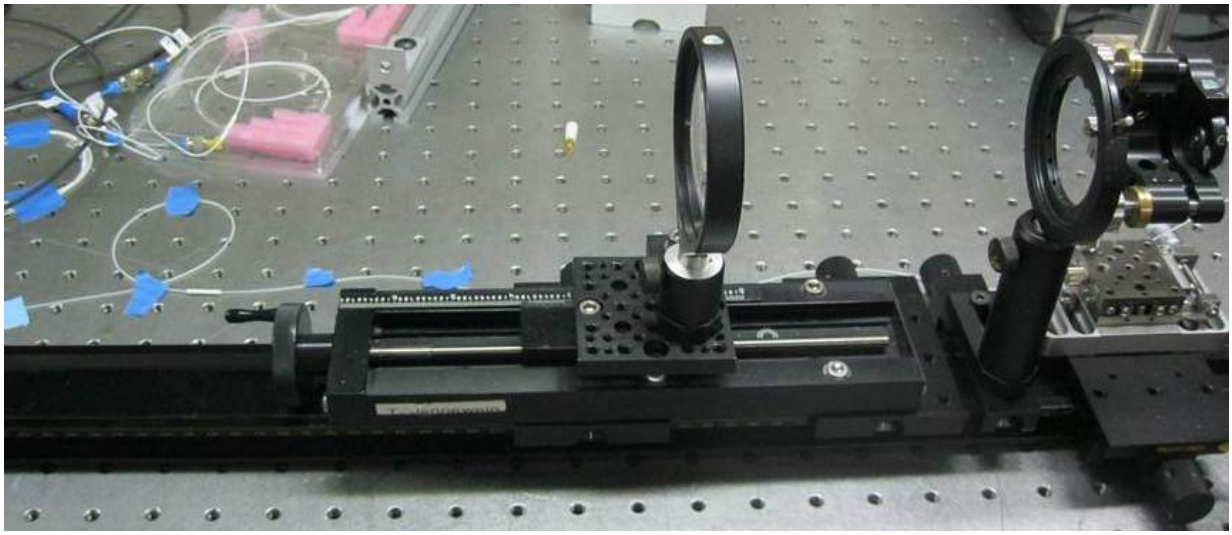

Figure 2.4: A lens mounted on a translation stage allows for control of the loss in the quantum channel by manipulation of the beam size. To the right, upconverted green photons exit a coupler to the single mode fiber carrying the output beam. The quantum receiver is located further away to the left.

$$|D\rangle = \frac{|H\rangle + |V\rangle}{\sqrt{2}} \tag{2.1}$$

Since the output intensity of the 532 nm photons depends linearly [66] on the input intensities of the 810 nm and 1550 nm photons, we can modulate the output beam by keeping the Ti:Sapph intensity stable and using fast telecom modulators (see section 2.2.1) to modify the polarization and intensity of the 1550 nm beam.

Figure 2.3 displays a system-level schematic of the WCP source as part of the overall decoy-state BB84 QKD system. Alice prepares a sequence of photon states via telecom modulation (figure 2.1) and upconversion (figure 2.2). The reference signal from the FPGA-controlled modulation system is time-stamped by the time-tagger unit. The resulting time-tags are transferred in real-time to Alice's computer over USB where they are analyzed (together with the ones from Bob) to produce the raw and sifted keys (see chapter 3).

Using two 90-10 fibre beamsplitters connected in series, Alice splits off and measures 1% of the green photons for source characterization, discards 9% and sends the remaining 90% of upconverted photons over the quantum channel to Bob. The loss in the quantum channel is controlled through a lens mounted on a translation stage (figure 2.4). The position of the lens determines the size of the light beam arriving at the quantum receiver (section 2.3). Bob detects a very small fraction of this beam. After the quantum transmission, Alice and

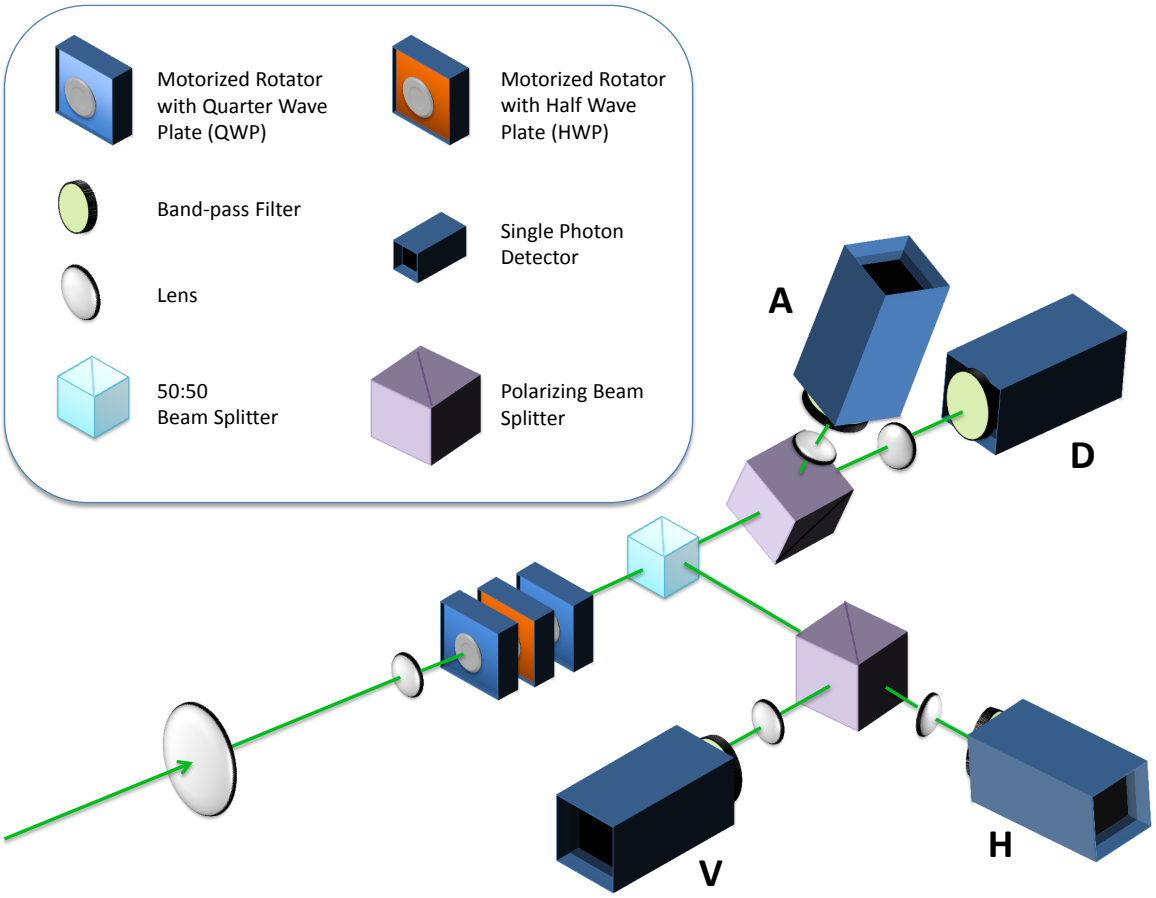

Figure 2.5: Schematic representation of our quantum receiver implementing measurement of the BB84 polarization states: horizontal (H), vertical (V), diagonal (D) and anti-diagonal (A). A wave plate triplet placed in motorized rotation stages is used for our automated polarization alignment procedure.

Bob communicate classically over the local area network (LAN) to complete the classical post-processing steps of the QKD protocol (see chapter 3).

2.3 Quantum Receiver

Once Alice has prepared and sent her quantum states with her WCP source as described in section 2.2, Bob must measure the transmitted qubits to complete the quantum part of the QKD protocol. Traditionally, QKD experiments implement Bob's measurement with optical elements arranged in a single plane on a breadboard. However, such configuration is not very suitable for a satellite payload as it is not very compact. As a proof of concept, our quantum receiver is built from commercially available components organized in a three-

dimensional robust frame. It features spectral filtering, passive basis choice, automated polarization alignment, and few moving parts. A design schematic of the receiver is shown in figure 2.5 and photographs are provided in figure 2.6.

The quantum receiver is built with several Thorlabs cage systems. Each consists of four 6 mm-thick stainless steel rods which form a rectangular box or *cage* along the direction of the optical axis. Each box is terminated on both sides by rigid square elements (with 30 mm or 60 mm sides) to which the rods attach. Optical components are mounted along this structure [67]. The receiver is made up of three subsystems attached together to a small (12 in $\times$ 6 in) breadboard:

- Telescope
- Polarization compensation unit
- Polarization measurement module (i.e. "Bob module")

The telescope collects a small portion of the uncollimated input beam coming from the source and reduces its diameter before passing it on to the polarization compensation unit. The telescope consists of two lenses located at the extremities of a 60 mm cage. The large lens has a 2 in diameters and a 250 mm focal length, while the small lens has a 6 mm diameter and a 10 mm focal length.

The polarization compensation unit consists of a wave plate triplet: a half wave plate (HWP) placed between two quarter wave plates (QWPs). The wave plates are mounted on motorized rotation stages which are controlled by a Newport XPS Universal High-Performance Motion Controller/Driver [68]. The triplet is employed in our automated polarization alignment procedure described in section 3.5. In brief, this set of wave plates can implement an arbitrary polarization correction and a change-of-basis for measuring circular polarizations.

Bob's polarization analysis module is used to measure incoming photons, and hence to decode the quantum information they carry. As illustrated in figure 2.5, the quantum state of each photon is collapsed to one of the four BB84 polarization states: horizontal (H), vertical (V), diagonal (D) and anti-diagonal (A).

In the BB84 protocol, Bob needs to select which basis, H/V or D/A, to use for each polarization measurement. Since we aim to have few moving parts, our receiver implements the basis choice passively with a non-polarizing pentaprism beam splitter. In addition to the standard two ports (which each receive $\approx 47.5\%$ of the input beam), this custom beam splitter has a third port which could potentially be used for beacon detection or for a source going in the reverse direction.

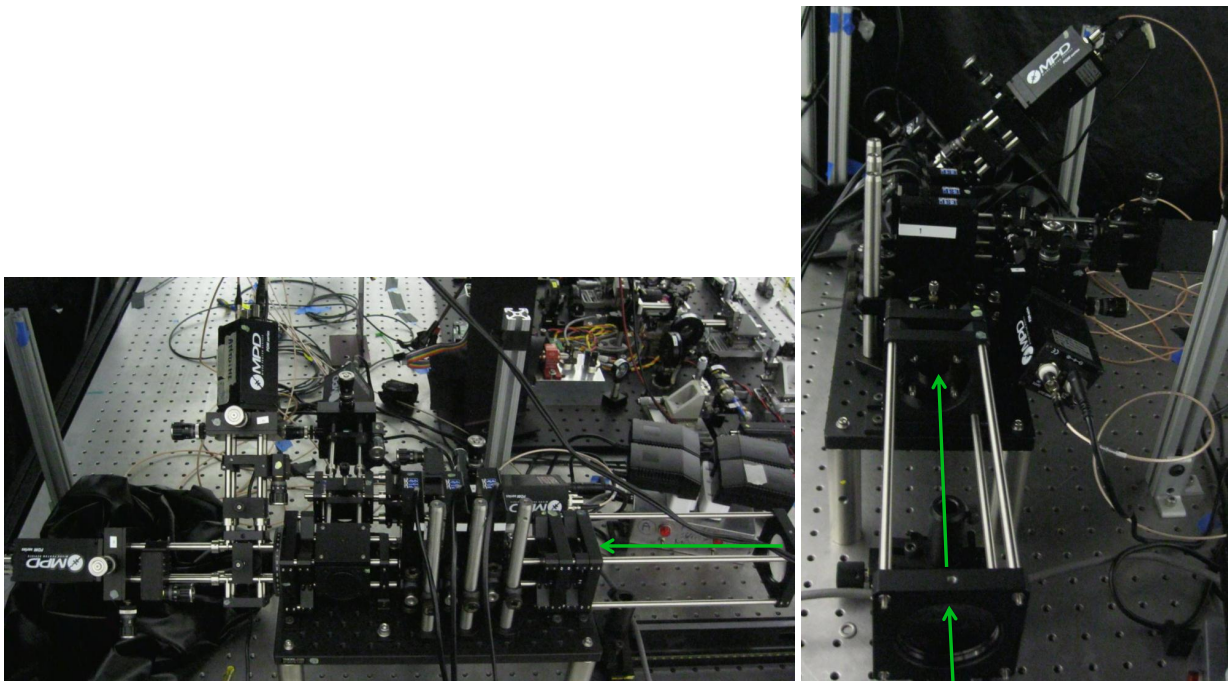

Figure 2.6: Photographs of the quantum receiver. The green arrows indicate the direction of the input beam which enters the receiver through the telescope. The beam then undergoes polarization compensation via the motorized wave plate triplet before it is analyzed by the passive basis choice "Bob module".

At each of the two main ports of the pentaprism, one for each measurement basis, there is a 30 mm cage system containing a 5 mm polarizing beam splitter cube (PBS) and a pair of perpendicular detection cage units. As shown in figure 2.6, to implement the measurement in the D/A basis, the cage system at the straight-through (primary) output of the pentaprism is rotated 45° around the optical axis. The measurement in the H/V basis is accomplished at the secondary output.

Each of the detection cage units is terminated with a narrow-band filter (for spectral filtering) and a thin silicon single photon avalanche diode (SPAD) module from Micro Photon Devices (MPD) [69] attached to an X-Y translation stage for precise position alignment. To focus the beam onto the 50 μm active area of the detector, a 50 mm focal length lens is placed in front of each detector. At the output wavelength (532 nm) of our WCP source, these detectors have a high figure of merit [65], $H > 10^8$, and hence they have high detection efficiency ($\approx 48\%$), low dark count rate (10-25 counts per second), and low timing jitter (<100 ps) allowing for precise temporal filtering (see section 3.4). Detection events from the SPAD modules are time-stamped with our fast time-tagging hardware (see section 3.1) which has a timing resolution of 76 ps. Each time stamp (denoting when the event happened) and channel information (indicating which detector was triggered) is transferred out of the time-tagger over a standard USB link and read in by the QKD software (see chapter 3).

Chapter 3

Data Acquisition and Processing

Our QKD system consists of two main sub-systems traditionally named Alice and Bob. In our setup, Alice is the "ground-station" component acting as the quantum source, while Bob is the "satellite" component acting as the quantum receiver. Data are collected by Alice and Bob via dedicated timing hardware (section 3.1). In addition to the time-tagging infrastructure, computing facilities on each side are required to implement a full QKD post-processing protocol (section 3.2). In a real system, there will be a ground-station server and a low-power embedded computer onboard the satellite. In our lab system, two standard x86-64 desktop computers are used on each side and the local area network (LAN) acts as a classical communication channel (section 3.3). However, the desktop computer on Bob's side is only used to record the experimental data, while all the satellite-side data processing is done by a low-power ARM-based board.

3.1 Data Acquisition Hardware

This project is done in partnership with DotFast Consulting, which is providing the hardware, firmware and driver software used for time-tagging of significant events (e.g. photon detections) with sub-nanosecond precision. Such precision is needed to correctly identify corresponding events from the source and receiver systems within a very narrow time window (coincidence window). Good resolution is required for temporal filtering, to ensure that a high signal-to-noise ratio for the quantum signal is obtained. The overall accuracy of the event timestamps, or *time-tags*, is mainly affected by the jitter in the photon detectors and the clock synchronization of the time-tagging devices at the source and the receiver.

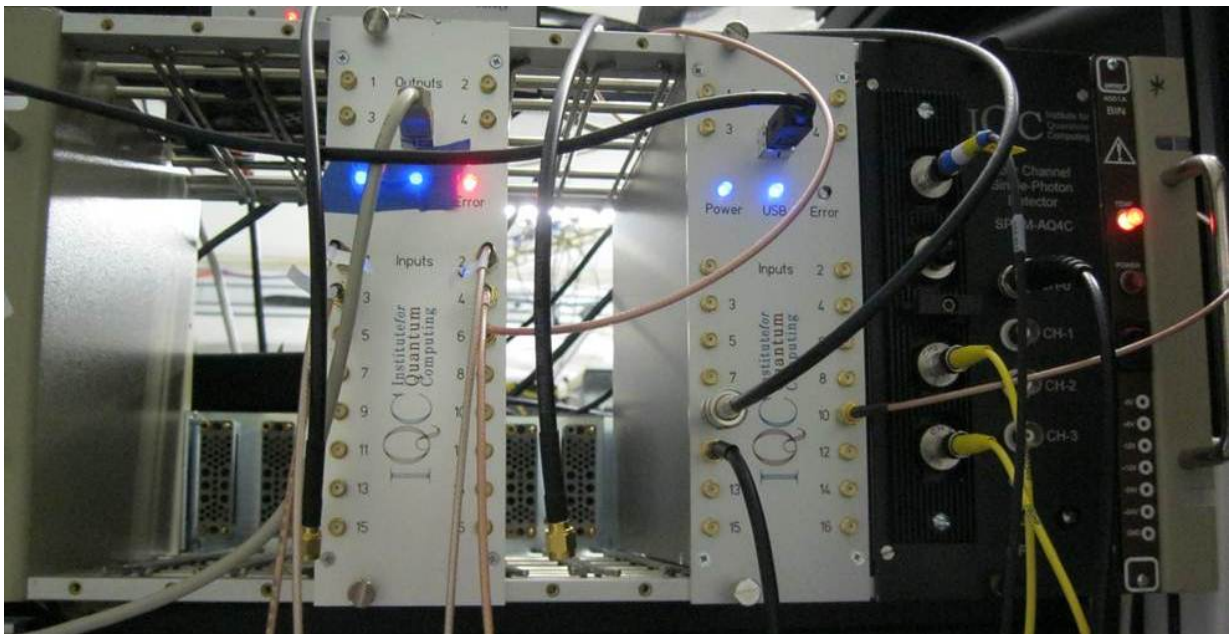

Figure 3.1: QKD data acquisition hardware in a NIM crate. Bob's time-tagger (on the left) attaches precise time-stamps to signals produced by the single-photon detectors in the quantum receiver (section 2.3) and the GPS unit. Alice's time-tagger (middle) is connected to the second GPS unit, the FPGA modulator (section 2.2) and the adjacent single-photon detector (SPD). Both time-taggers output their data streams (towards Alice or Bob's computer) via a standard USB link. On the right is a four-channel SPD used to detect characterization photons split-off from the source as shown in figure 2.3.

3.2 Post-Processing Protocol Overview

The *quantum* part of our QKD scheme ends when the source and the receiver finish exchanging quantum signals in the form of polarized photons (see section 1.2). The rest of the QKD protocol is purely classical and is done in software. The software is designed to perform the following general steps [1]:

1. Storage of time-tags and corresponding measurement results.
2. Exchange of timing information between Alice and Bob and extraction of coincident photon detection events (coincidences). This step produces the *raw key*.
3. Exchange of basis information for each coincidence and sifting down to only those events where the same polarization basis choice was made. The *sifted key* is produced. This step and the previous are performed by the coincidence analysis algorithm, which is described in detail in section 3.4.
4. Execution of a one-way error correction algorithm based on low density parity check (LDPC) codes [70]. Bob computes his syndrome information and sends it to Alice who performs belief propagation decoding [71, 72]. The *error-corrected key* is produced. The LDPC error-correction algorithm is discussed in section 3.6.
5. Estimation of the quantum bit-error ratio (QBER) of the channel and execution of Toeplitz matrix based [73] privacy amplification (see section 3.7) is performed by Alice and Bob. This final step gives the *final key*.

3.3 Software Design

The design of the software follows the general rule that Alice must perform as many of the computation-intensive tasks as possible, since the ground station can be made rich in computing resources with relative ease. Hence, Alice must execute the timing analysis, coincidence searching, sifting and belief propagation decoding algorithms. Alice is also responsible for time-tag readout at a very high rate (the source rate, as seen in section 2.2, is very high in an uplink scenario), as well as real-time display of statistics from the experiment.

The design of our data acquisition and processing system is depicted in figure 3.2. It consists of two time-tagger units, two x86-64 computers and an ARM board connected via the local-area network (LAN). Each time-tagger unit is connected to the 10 MHz and one

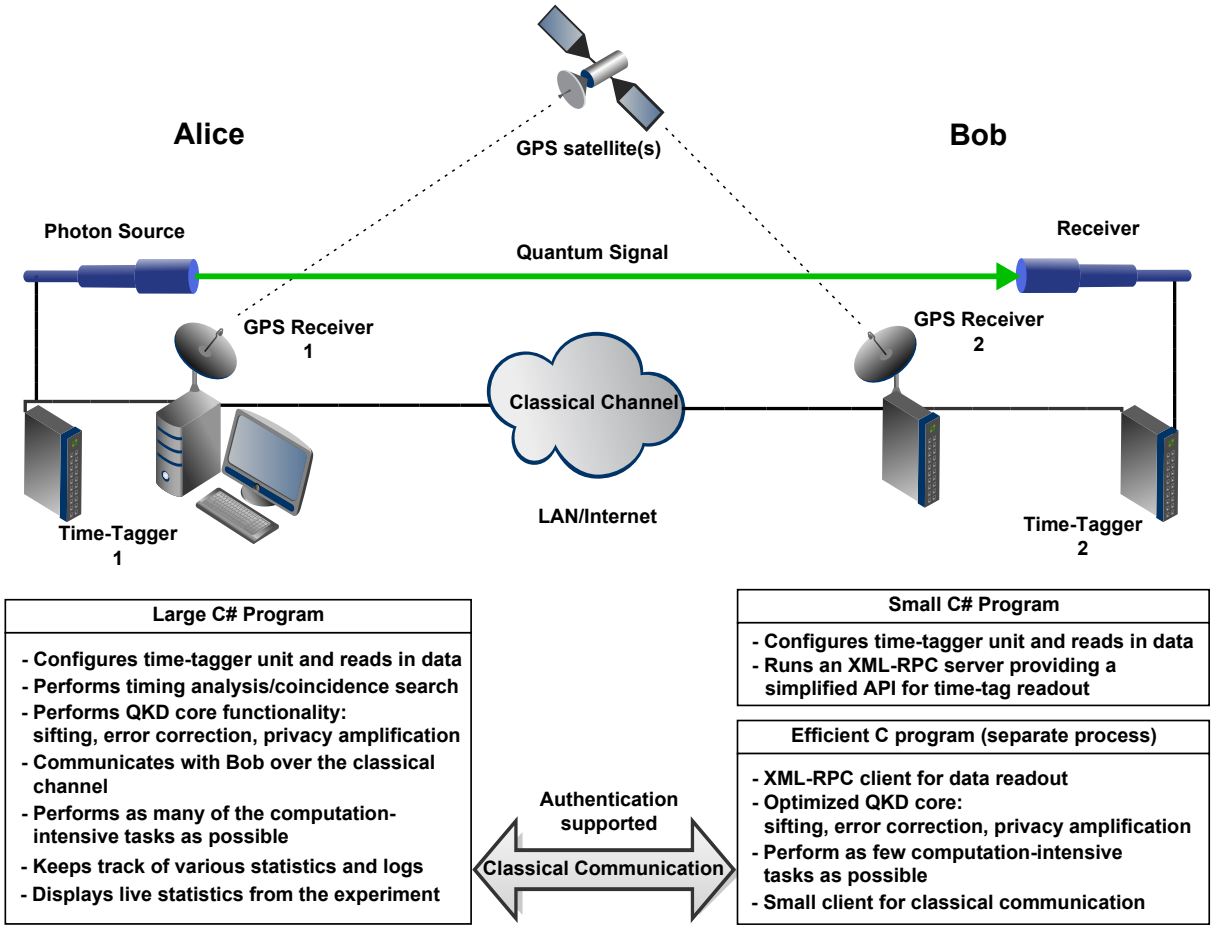

Figure 3.2: High-level software design. Alice's software (which runs on relatively powerful hardware) consists of an integrated solution written in C#, designed to perform as much as possible the computationally intensive operations of the QKD protocol. Bob's software (which runs on modest low-power hardware) consists of a small C# layer handling platform-dependent operations, acting in unison with an efficient C program that performs the necessary portions of the QKD protocol at the receiver side.

pulse-per-second (1 PPS) signals coming from a GPS receiver. The signals from the source are connected to Alice's time-tagger and the four outputs of Bob's detectors are connected to Bob's time-tagger.

Alice's software consists of a C# program originally based upon a software package written by Chris Erven at IQC, which has been significantly modified/rewritten and greatly expanded. It is currently capable of performing data recording, automated polarization alignment (section 3.5), coincidence and sifting algorithms (section 3.4), LDPC error correction (section 3.6), privacy amplification (section 3.7) as well as displaying live statistics from the experiment.

Bob's software is separated into two components: a driving control environment and an embedded component. The driving control component is written in C#. It is responsible for all platform-dependent tasks, e.g. loading Windows time-tagger drivers, configuring time-taggers, reading out time-tags and displaying live statistics in Windows widgets. With our setup, we aim to simulate/estimate in some way the limited-resource environment on the satellite. The actual satellite design is not yet established, so Bob's embedded software component needs to be implemented in a platform-independent way. It can then be executed as a separate entity on the x86-64 desktop computer or on our low-power ARM developer board. The embedded component is written entirely in C, which will allow it to be easily ported to the platform of choice for the satellite mission. Ideally, we would have preferred to use a single embedded system for Bob. However, the lack of Linux drivers for the time-tagging hardware led us to adopt the present design. For better abstraction and ease of implementation, the embedded component communicates with the environment via remote-procedure calls (RPC). If needed, the RPC layer can be replaced by a more appropriate mechanism, once the satellite bus design is finalized.

With this setup, we have been able to test the embedded code on an ARM-based developer board running a basic version of Linux. Since Bob's embedded component runs in a standalone process, its usage of computing resources can be accurately monitored. Moreover, the driving control environment component can keep track of the bandwidth used for classical communication. The implementation of Bob's software has been completed up to and including the privacy-amplification stage of the protocol. We use this implementation to guide our analysis of the computing requirements of Bob's part of the QKD protocol. Those are presented and thoroughly discussed in chapter 4.

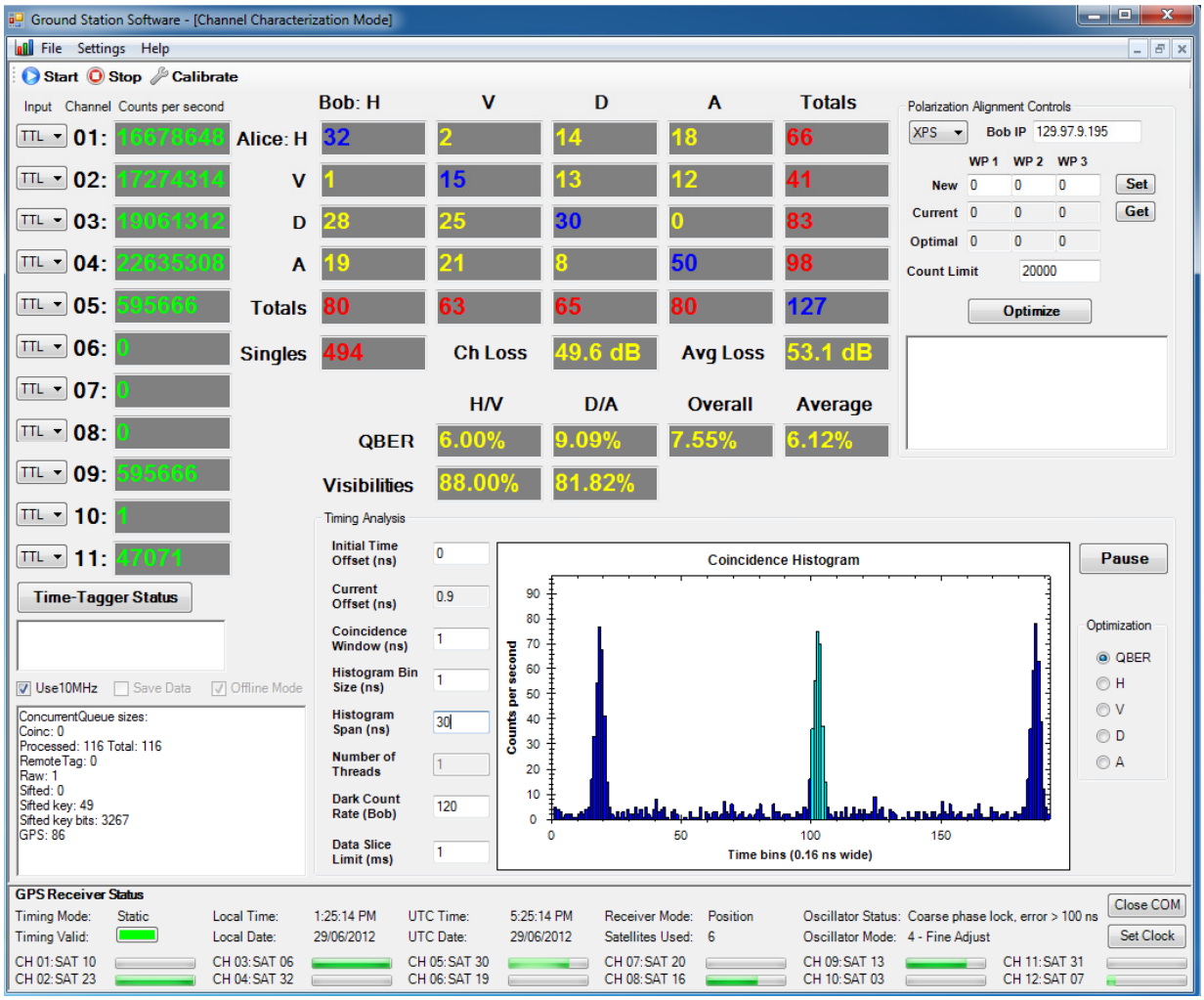

Figure 3.3: Screenshot of the software control panel at Alice (ground). Data rates at different channels of the time-tagger unit are displayed on the left, along with the time-tagger status. The 4×4 matrix in the top middle displays the observed state characterization; the numbers in blue represent the number of events when Alice and Bob's state measurement and basis choice coincide. The automated polarization alignment controls are located in the top right corner. Below the running statistics such as QBER and channel loss is the timing analysis panel, which displays the resulting coincidence histograms.

Time-tag/Character	Channel
⋮	⋮
1	27
9	27
5	27
7	27
1	27
3	27
.	27
17963731085	4
0	27
0	27
,	27
4	27
3	27
17983901285	2
2	27
8	27
.	27
7	27
4	27
3	27
18004666905	3
8	27
18007903317	1
,	27
N	27
,	27
⋮	⋮

Table 3.1: Sample data stream, collected from the experimental apparatus, containing interleaved photon detection time-tags and serial messages. Serial messages appear on channel 27, while channels 1 to 4 correspond to photon detections. Here, the data stream contains a section of the message: $GPGGA,195713.00,4328.7438,N,08033.2876,W,0,07,00.0,00314.1,M,,,,*12 in the National Marine Electronics Association (NMEA) 0183 message format

3.4 Timing Analysis and Coincidence List Generation

Let us focus our attention to the problem of timing alignment of photon detection signals. In our case, the source is producing quantum signals at rates of up to 76 MHz, while the detection rates at the receiver usually vary from a few hundred counts per second up to about 100 kHz depending on the loss in the quantum channel. The goal is to identify exactly which photon emitted from the source corresponds to a given detection event at the receiver. This task poses some practical challenges:

- *Clock Rate Synchronization*: Alice and Bob's time-tagger clocks might not count time at the same rate, leading to timing drifts.
- *Frame Synchronization*: Both time-taggers don't start counting time at the exact same instance, so Alice and Bob's timing frames have an initial fixed offset.
- *High Rate Periodic Source*: The data acquisition hardware cannot operate at the Ti:Sapph laser's 76 MHz output rate.
- *Variation in the Photons' Time-of-Flight (TOF)*: In the lab, the photons' TOF is always constant, but in a real system the TOF will be continuously changing due to the fast motion of the satellite (see figure 3.4).

To alleviate the unfeasibly heavy load on the time-tagger at the source, only one in 128 Ti:Sapph laser pulses is tagged. The signals in between are assumed to have been emitted at regular intervals, that is, we assume that the laser's period is stable for about 1.664 μs. As discussed in section 2.2, the desired polarization and signal/decoy state of each pulse in the known (only to Alice) sequence of 128 pulses is produced by the modulator.

For clock rate synchronization, the time-tagging units' internal clocks are aligned to a time-base signal of 10 MHz provided by a GPS receiver at each side. Frame synchronization is achieved with the 1 PPS signal provided by the GPS receivers. Each GPS receiver also supplies position data, which is used in conjunction with the time information to estimate the distance between Alice and Bob, and hence, the TOF of the photons between the source and the receiver.

As shown in table 3.1, the GPS time and position information is recorded within the time-tagging system, thereby locking all time measurements to the time base of the time-tagging system. This is important because the GPS data packet, containing position, speed and direction of motion, and the corresponding time-stamp can be used to improve the

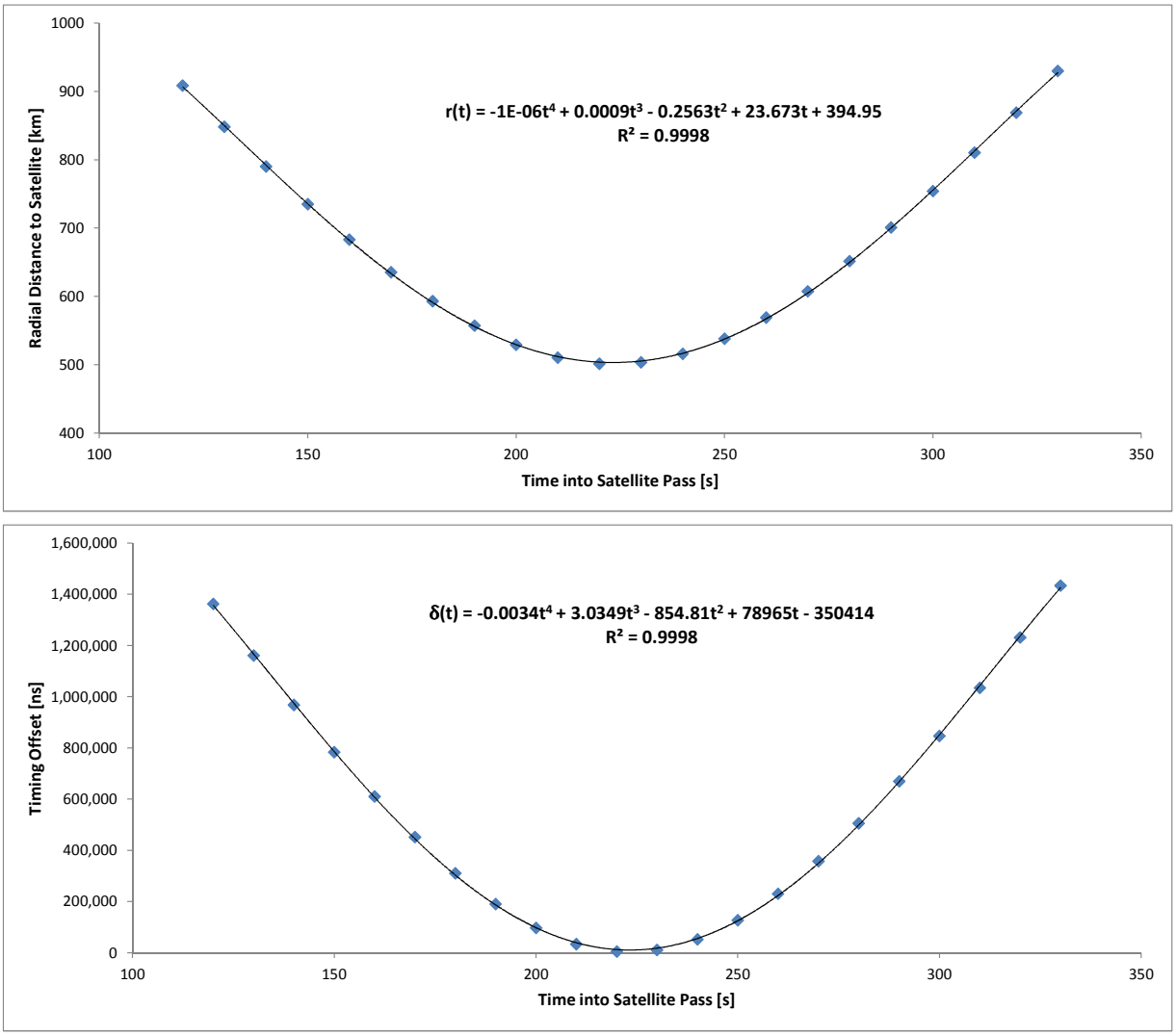

Figure 3.4: Radial distance $r(t)$ versus time plot (top) and corresponding timing offset function $\delta(t)$ (bottom) for a "best" duration satellite pass [18] over Ottawa. Based on the satellite's radial distance data (top), the TOF of photons relative to an arbitrarily chosen reference point (e.g. $r_0 = 500$ km) gives the relative timing offset at each data point. A smooth timing offset function $\delta(t)$ is obtained using a high-order polynomial fit (as shown here), or an even more precise satellite-orbit model. An offset $\delta(t_i)$ can then be added to each time-tag t_i coming from Bob's time-tagger.

orbit prediction, and hence, to calculate the TOF of photons to an error of at most 100 ns. The main source of error is the jitter in 1 PPS signals coming from the GPS receiver.

Moreover, timing signals sent over a beacon laser to the satellite can be recorded on the time-tagging system on the ground and the satellite, and provide additional information about the TOF of optical signals. It is expected that this optical time-transfer can be accurate to about 50 ns. Depending on the power budget and the bandwidth of the beacon link, this accuracy might be significantly better (less than 1 ns).

3.4.1 Coincidence Algorithm

In our current lab system, photon detections are accurate to 78 ps, but 1 PPS signals are accurate to 100 ns. Thus, additional analysis is required to identify corresponding emission and detection events to within a desired coincidence window of about 0.3 ns to 2 ns. The algorithm for alignment of photon detections at the satellite receiver with signals sent from the ground source is a computationally expensive task. It is therefore best to assign it to Alice on the ground. The algorithm is based on the timing information from Bob's time-tags, Alice's transmitted photon states, Alice and Bob's GPS timing and position data, as well as a small subset (≈5%) of Bob's measured outcomes. Sampling Bob's outcomes is necessary to identify the correct offset, because the WCP source currently employed is periodic with a period of approximately 13 ns (as mentioned, the 1 PPS signals are accurate to 100 ns).

Our implementation employs a histogram-based optimizing coincidence search within a predefined time range (about 100 ns wide). Moreover, the information from the sampled outcomes also comes in handy during the error correction stage of the QKD protocol (see section 3.6). As discussed in section 1.3, all detection events are stored on the satellite while it flies over an optical station and are only downloaded to the ground station server when the satellite flies over an RF station. It is therefore essential that the coincidence algorithm is executed in parallel on multiple frames, each frame containing a second's worth of data. The challenge becomes to distribute the data evenly among multiple computing tasks/threads, without disrupting the temporal order of the frames. Our solution is to assign an index to each frame:

$$I_{frame} = UTC_{second} \pmod{N_q} \tag{3.1}$$

Here, UTC_{second} is the "seconds" field of the Coordinated Universal Time (UTC) time-stamp contained in the GPS data packet and N_q is the total number of processing queues on the ground. Each queue is serviced by a C# Task [74], the CoincidenceTask, which in

our case is mapped to a physical thread by the operating system (Windows 7). Since both Alice and Bob have access to the GPS data packet, they can easily compute I_{frame} for each frame and thus establish the correct ordering. Alice processes the frames in parallel as soon as they arrive, while Bob requests the results from the queues in a round-robin fashion. Each coincidence task

1. Generates a coincidence histogram as shown in algorithm 3.1.
2. Finds the correct peak from the histogram. As seen in figure 3.5, coincidence peaks appear every 13 ns, matching the period of the Ti:Sapph laser. Only one peak has the correct offset and QBER estimates from the previous step are needed to identify it. This procedure is outlined in algorithm 3.2
3. Performs basis sifting. With the unbiased BB84 protocol (see section 1.2), about half of the coincident events are discarded due to basis mismatch.
4. Generates the coincidence lists for Alice and Bob. Each list contains indices of the emission/detection events which make up the sifted key. Bob's coincidence list is enqueued on the correct output queue. Bob can then receive the list over the classical channel.
5. Performs decoy-state analysis and parameter estimation.

Algorithm Complexity

Even though the coincidence algorithm is performed on the ground, it needs to be sufficiently efficient so that timing data is processed as soon as it is downloaded from the satellite. Recall that all post-processing has to be accomplished within the duration of the satellite pass over an RF station. The scalable parallelization (as discussed earlier) of the coincidence algorithm ensures that its runtime can be reduced by simply adding more CPU cores.

It is also important to analyze the overall complexity of processing one frame of data. From algorithms 3.1 and 3.2, it can be seen that the runtime is linear in the number of Bob's time-tags. To show this result, let us define the following parameters treated as constant factors:

- h_s = histogram span [seconds]
- h_b = histogram bin size [seconds]

Algorithm 3.1: Coincidence histogram generation

Data: Two arrays of of timing and basis information, corresponding GPS data packets and "1 PPS" signals for Alice and Bob

Result: A coincidence histogram, each column/bin storing a list of coincidences, (emission, detection) event pairs, and estimated QBER

```
1  startShift = AlicePPS − BobPPS
   /* the histogram span is the size of the time window covered by the
      entire histogram, much larger than the coincidence window        */
2  minShift = startShift − histogramSpan
3  maxShift = startShift + histogramSpan
4  foreach time-tag b_i in Bob's time-tag array b do
       /* a_0 is the first time-tag in Alice's time-tag array a        */
5      minIndex = (b_i + minShift − a_0) / sequencePeriodMax
6      j = minIndex
7      while all possible offsets for b_i are accounted for in the histogram do
8          for k from 0 to sequenceLength do
9              AliceTag = a_j + k ∗ sequencePeriod / sequenceLength
10             shift = AliceTag − b_i
11             if shift < minShift then
                   /* shift is outside the left (negative offset) boundary of
                      the histogram                                    */
12                 k = k + (minShift − shift) / (sequencePeriod / sequenceLength)
13             end
14             else if shift > maxShift then
                   /* shift is outside the right (positive offset) boundary
                      of the histogram                                 */
15                 break (while loop)
16             end
17             else
                   /* update histogram bin (coincidence matrix, QBER
                      estimate, coincidence list) for that offset      */
18                 UpdateHistogram (shift, i, k)
19             end
20         end
21     end
22 end
```

Algorithm 3.2: Coincidence peak finding

Data: A coincidence histogram as produced by algorithm 3.1, the size of the coincidence window

Result: A subset of the histogram bins forming the correct coincidence peak, in the form of a HistogramPeak object (*optimalPeak*)

```
/* Group consecutive histogram bins based on the coincidence window to
   create all possible HistogramPeak objects.  At creation, the offset
   of a HistogramPeak is computed from a weighted average of the bin
   counts                                                              */
peakSize = coinWindow / histogramBinSize
for i = 0 to (histogramLength − peakSize) do
    peakList.Add(HistogramPeak(i, peakSize, histogram))
end
/* Sort the list of HistogramPeak objects by number of coincidences */
peakList.Sort()
topFraction = coinWindow / TiSapphPeriod
/* The first (topFraction ∗ peakList.Length) HistogramPeak objects in
   the sorted peakList now contain the TiSapph peaks which repeat
   periodically as seen in figure 3.5                                  */
optimalPeak = peakList[0]
minQBER = 1
for i = 1 to (topFraction ∗ peakList.Length) do
    if peakList[i].qber < minQBER then
        /* The optimal peak is the one with the smallest QBER         */
        optimalPeak = peakList[i]
        minQBER = peakList[i].qber
    end
end
return optimalPeak
```

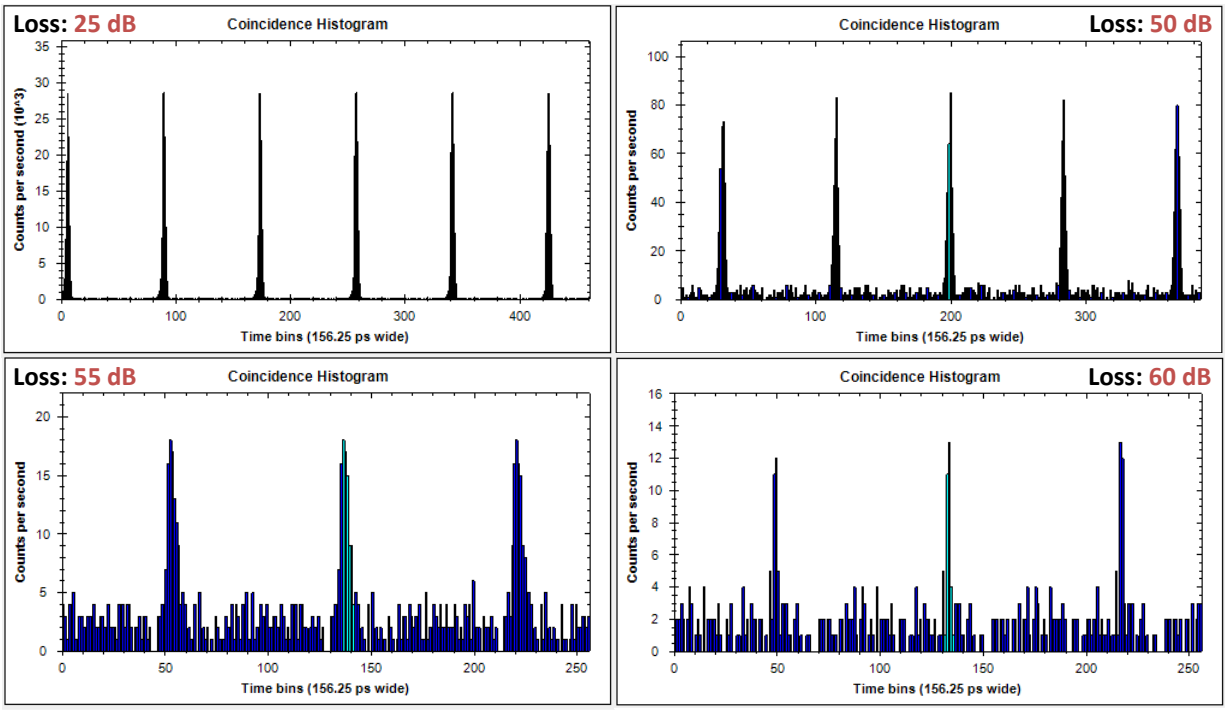

Figure 3.5: Coincidence histograms at different channel loss levels. Each column in a histogram represents the number of time-tags having a given timing offset. Note the significant drop of SNR as the loss in the channel increases and detection events due to background light become more and more pronounced. Nevertheless, even at 60 dB, the coincidence algorithm is able to correctly identify the peak (in light blue). This is an example of *temporal filtering*.

- w = coincidence window size [seconds]
- N_b = number of detection events at the receiver (Bob) per one second frame
- L_s = length of repeating pseudo-random sequence
- T_{laser} = Period of Ti:Sapph laser

Then the complexity of algorithm 3.1 (step 1)

$$\Theta\left(L_s \left\lfloor \frac{h_s}{T_{laser}} \right\rfloor N_b\right) \tag{3.2}$$

The complexity of algorithm 3.2 (step 2) is

$$\Theta\left(\left\lfloor \frac{w}{h_b} \right\rfloor \left\lfloor \frac{h_s}{h_b} \right\rfloor \log \left\lfloor \frac{h_s}{h_b} \right\rfloor\right) \tag{3.3}$$

Steps 3 and 4 consist of simply iterating through all ($\leq N_b$) coincidence pairs, so the complexity is $\Theta(N_b)$. Finally, step 5 takes constant time.

Thus, the overall complexity (per frame of data) of the coincidence algorithm is $\Theta(N_b)$. Multiple frames are processed in parallel depending on the number of available CPU cores.

3.5 Automated Polarization Alignment

Author Contributions

Brendon Higgins developed the polarization analysis (utilizing existing quantum state tomography code written by Nathan Langford) and compensation optimization protocol. With Brendon's help, I implemented the motorized stage control software, the alignment control sequence and time-tag mode synchronization, thereby integrating the automated polarization alignment procedure into the QKD software.

3.5.1 Overview

As discussed in section 1.2 our QKD scheme employs photon polarization to encode quantum information. A practical problem arises from the fact that the quantum source (section 2.2) and receiver (section 2.3) do not necessarily share the same polarization reference frame. Earth's atmosphere itself is not birefringent [75, 76] (i.e. it does not substantially

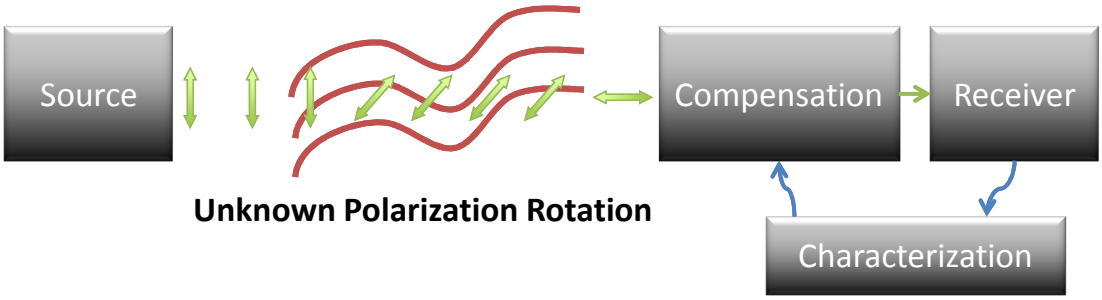

Figure 3.6: Schematic of the automated polarization alignment procedure. The source produces known polarization states which are then subject to an unknown rotation. The rotation is characterized through measurements of the received photons. Appropriate compensation is then applied to correctly align the polarization reference frames of the source and the receiver.

modify photon polarization), however the many optical fibers which are used to guide light throughout the source can significantly alter photon polarization. Moreover, fiber birefringence is affected by temperature [77], so any change in the lab temperature ultimately affects the alignment between the source and the receiver. Thus, continuous operation of the experiment necessitates regular "tuning" of the alignment.

To resolve this problem, we have developed an automated procedure to characterize and compensate the relative rotation between the two polarization reference frames, as depicted in figure 3.6. Unlike other implementations which employ an independent strong laser signal [78], our protocol relies solely on the photon detection statistics of the known sequence of states prepared by the modulator (as described in section 2.2).

3.5.2 Characterization

The characterization step essentially reduces to *quantum state tomography* of photon polarization qubits [79], a common tool in experimental quantum mechanics and quantum information processing. Our goal is to best estimate the unitary operator corresponding to the unknown polarization rotation. The unknown rotation can be represented as an arbitrary SU(2) unitary U of the form:

$$U = \begin{bmatrix} e^{i\alpha}\cos\theta & e^{i\beta}\sin\theta \\ -e^{-i\beta}\sin\theta & e^{-i\alpha}\cos\theta \end{bmatrix} \tag{3.4}$$

If the input polarization state (in density operator form) is ρ, then the state produced by

the action of the unknown rotation is given by

$$\rho' = U\rho U^\dagger \tag{3.5}$$

As discussed in section 2.2, the source prepares a sequence of known polarization states, so we have full knowledge of ρ. The idea is to perform measurements on ρ' in order to estimate the real-valued angles α, β, and θ which fully describe U. This approximation is accomplished with maximum likelihood estimation on measurement statistics from an *overcomplete* set of outcomes: horizontal (H), vertical (V), diagonal (D, $+45°$ from horizontal), anti-diagonal (A, $-45°$ from horizontal), right-circular (R), and left-circular (L) polarizations.

3.5.3 Compensation

Once the unitary is characterized, a set of compensation optics (see section 3.5.4) is used to implement the inverse of the unitary, thereby canceling out the relative rotation. Similar to the unknown polarization rotation U, the compensation operation can be represented as an SU(2) unitary operator, C. Taking into account the action of C, the measured photon polarization state becomes

$$\rho' = CU\rho U^\dagger C^\dagger \tag{3.6}$$

Hence, we want to find C which satisfies $CU \approx I$, the identity rotation. If this condition is met, the output state ρ' will closely approximate the original input state ρ.

3.5.4 Experimental Implementation

To implement this two-step protocol experimentally, we need to

- Perform measurements in the circular basis, as required for the characterization step (section 3.5.2). This functionality is not readily provided by the polarization analysis module in the quantum receiver (section 2.3).
- Compensate for any given polarization rotation, as required for the compensation step (section 3.5.3).

Both of the above requirements can be fulfilled by a small set of wave plates. In particular, we employ a sequence of quarter-wave plate (QWP), half-wave plate (HWP), and QWP. As shown in figure 2.5, the wave plate triplet is placed in motorized rotation stages and

located at the quantum receiver, just prior to the measurement. This triplet can implement an arbitrary polarization rotation and can also be used to achieve a change-of-basis for measuring circular polarizations.

With this setup, the characterization process is done in two stages: linear polarization measurements are taken first, followed by measurements in the circular basis. The compensation operation C is fully determined by the rotation angles of the three wave plates: ϕ_1, ϕ_2, ϕ_3. We estimate the angles with an optimizing search over a range of values for ϕ_1, ϕ_2, ϕ_3, where we maximize the fidelity between ρ and ρ'. The alignment optimizer is implemented as a stand-alone executable written in C. The QKD software invokes this analysis program, providing measurement statistics, and obtaining optimal wave plate angles.

To preserve the correct binning of measurement outcomes, we designed a simple set of modes which Bob applies as special bit-masks to each time-tag. Alice then interprets the bit-masks and displays the measurement statistics accordingly. Those time-tag modes are:

- QKD mode: the standard operation mode of the experiment.
- POLN_LINEAR mode: polarization alignment in progress, measurements of linear polarizations.
- POLN_CIRCULAR mode: polarization alignment in progress, measurements of circular polarizations.

Below is the full alignment control sequence from Alice's point of view. She is the one in control of the protocol, while Bob is programmed to simply follow her commands.

1. Turn the wave plates to their calibrated optic axis positions, in order to measure linear polarizations.
2. Send a message telling Bob that the experiment is in POLN_LINEAR mode.
3. Wait until the CoincidenceTask collects enough data in this mode.
4. Turn the wave plates for measurement of circular polarizations. This is accomplished by rotating the second QWP by 45°.
5. Send a message telling Bob that the experiment is in POLN_CIRCULAR mode.
6. Wait until the CoincidenceTask collects enough data in this mode. When done, go back to QKD mode

7. All the counts are summed up and the stand-alone optimizer is invoked. If the optimizer succeeds, set the wave plates to their optimal values; otherwise, revert to their original values.

8. Send a message telling Bob that the experiment is back to the regular QKD mode.

It is also important to exclude detection events recorded while the wave-plates are rotating. For this purpose, the stage control task at Bob sends messages to the data acquisition task telling it when the wave-plates started and stopped moving, and a special INVALID mask is applied to the time-tags during rotation.

Note that only the measurement and motorized stage rotation is performed at the receiver end. Bob only has to transmit his measurement statistics (which reduce to a 24 32-bit integers) to Alice on the ground. There, it is processed as outlined above and the compensation settings (ϕ_1, ϕ_2, ϕ_3) are transmitted back to the receiver.

Integrating this automated alignment procedure into the QKD software has significantly improved our ability to achieve low QBER at various loss regimes. Manual alignment is close to impossible at high loss due to the low SNR and high sensitivity to background light. Furthermore, some form of automated polarization alignment will be crucial in future experiments with moving receiver (eventually on a satellite), since in those cases, the orientation of the reference frames as well as the loss in the quantum channel will be varying with time.

3.6 Error Correction

The next step in the QKD post-processing protocol (section 3.2) is error correction. The broad topic of coding theory is a field in itself and has been studied extensively, especially in the classical world. This section provides a brief overview of error correction in QKD and focuses on our specific implementation employing low-density parity-check (LDPC) codes. We found that LDPC codes are very suitable for satellite-based QKD due to low communication overhead and the inherent asymmetry in the amount of processing required at each side.

3.6.1 Overview

After transmission of quantum signals over the free-space quantum channel, Alice and Bob establish their respective raw keys. Those keys are not identical as they contain

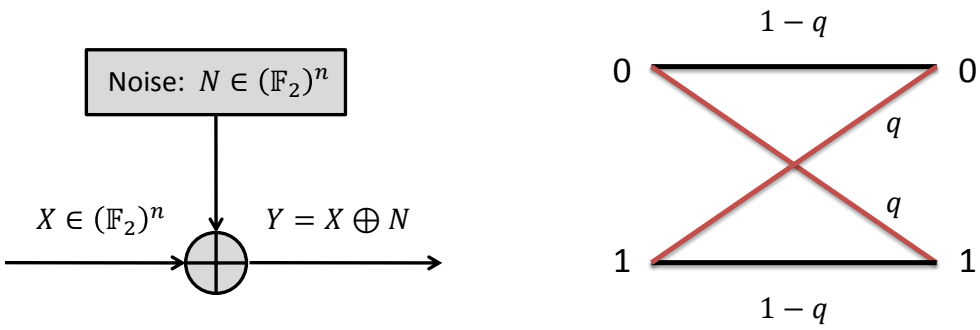

Figure 3.7: Schematic representations of the binary symmetric channel (BSC). The noise vector N follows a Bernoulli$(1-q, q)$ distribution. As shown on the right, q is the probability of a *bit-flip error*, i.e. the crossover probability. In our error correction model, Bob on the satellite is assumed to have the error-free sifted key X, while Alice on the ground holds the "noisy" sifted key Y which needs to be corrected to match X (based on some limited information about X).

several types of errors. Intrinsic errors due to basis mismatch are eliminated by the sifting procedure which in our implementation is incorporated into the coincidence algorithm (section 3.4). Without the presence of an eavesdropper, the remaining errors are caused by the unavoidable imperfections in the physical realization of the quantum source, channel and measurement. Some examples of these are:

- Imperfections in the state preparation procedure at the source (section 2.2). Small fluctuations in the laser spectra and modulator voltage settings as well as room temperature variations cause the source to sometimes produce incorrect states.
- Polarization reference frame misalignment is only approximately corrected with the automated polarization algorithm (section 3.5).
- At the receiver, imperfect polarizing beam splitters, detector dark counts and stray background light can lead to accidental coincidences.

An eavesdropper can also cause discrepancies in the key, however, Alice and Bob have no way of distinguishing between different error sources. For security reasons, they must assume that all key disparities are due to eavesdropping. Note that in order to compensate for key errors, it is not necessary to perform *quantum* error correction [80, 81], even though such a scheme (albeit impractical in this case) could potentially be implemented. Alternatively, the communication channel[1] is modelled as a classical binary symmetric channel

[1]Here, the term *communication channel* does not refer to the physical medium for information transmission, but rather to the information theoretic statistical model consisting of input, $\{x\}$, and output,

(BSC) [82] with crossover probability $q = QBER$ as shown in figure 3.7. An important difference from classical coding theory is that the transmitter does not encode information in "codewords" which (in a classical scenario) are typically transmitted over the noisy channel and decoded at the receiver. Instead, we assume that Bob (the receiver) already holds a "correct" version of the key, while Alice (the transmitter) attempts to reconcile her key. During this key reconciliation process, partial information about Bob's key is revealed over the classical channel.

Nevertheless, Shannon's channel coding theorem [83], also known as the noisy-channel coding theorem, still applies in this context. It sets a fundamental upper bound on the key rate as well as a lower bound on the amount of information (about Bob's key) which needs to be disclosed in order to correct the key. Error correcting codes are generally ranked based on their *efficiency*, denoted $\eta_{\text{EC}}(q)$, which specifies how well they perform relative to the *channel capacity*, $C = \max_{p(x)} I(X : Y)$. For example, a value of $\eta_{\text{EC}}(q) = 1$ indicates that the code can achieve the optimal key rate (i.e. the code operates at capacity), while $\eta_{\text{EC}}(q) > 1$ implies that the code rate is below the Shannon limit and additional information needs to be leaked out to successfully correct the key. Shannon's coding theorem also tells us that a value of $\eta_{\text{EC}}(q) < 1$ is unfeasible. Apart from their efficiency, error correcting codes are also evaluated based on their classical communication requirements (which is generally related to their interactivity) and their computational complexity for each party.

The most common error correction algorithm employed in QKD until recently has been the Cascade protocol. It is first proposed in an early form in [84] and fully developed by Brassard and Salvail [85]. Further optimizations are explored in [86, 87]. Cascade is an iterative, interactive reconciliation protocol. At each round, Alice and Bob divide up their sifted key into blocks (the choice of block length has been the main area of optimization). They proceed to compute and communicate their block parities. In the case of a parity mismatch—implying an odd number of errors—they recursively perform binary search on smaller and smaller blocks to locate one error, all along exchanging parities over the classical channel. With that error corrected, subsequent rounds may uncover further disparities. This cascading search is the underlying reason for the *Cascade* name. The number of iterations is chosen in such a way as to keep the probability of residual errors below a small threshold [85]; a number of rounds under 20 is usually sufficient.

The optimized Cascade protocol has an acceptable efficiency in the range $(1.14, 1.22]$, dependent on the QBER [88]. However, it suffers from several other shortcomings which make it impractical for long-distance links with limited classical communication capabilities, as is the case with satellite-based QKD. As mentioned, Cascade is very interactive,

$\{y\}$, alphabets and a transition probability matrix $p(y|x)$ for each element of the two.

so its real-world performance is strongly dependent on the latency of the classical channel. To make matters worse, the number of required interactions increases with the QBER and so does $\eta_{\mathrm{EC}}(q)$ as seen in figure 4 in [88]. Furthermore, Cascade is a symmetric procedure, i.e. both parties go through the same steps, so it requires equal amount of data processing resources at each end. This symmetry is not an obstacle in ground implementations, but may be problematic for satellite links.

In an effort to do away with some disadvantages of Cascade, Buttler et al. [89] proposed Winnow—a different reconciliation algorithm which attempts to apply techniques from coding theory in the context of QKD. Winnow is significantly less interactive than Cascade as it eliminates the binary search step. Similarly to Cascade, both parties divide their key into blocks and exchange the parities. In the case of a parity mismatch, instead of binary search, Alice and Bob compute Hamming's syndromes [82] which are used to identify the location of an error within a block. Some optimizations relating to the block length choice are discussed in [90]. Even though Winnow eases the bandwidth requirement for the classical channel, it suffers from worse efficiency $\eta_{\mathrm{EC}}(q)$ compared to Cascade.

More recently, QKD commercialization has led to increasingly higher key rates and a pursuit for faster and more efficient error correction. A powerful tool in the modern coding theory toolbox, forward error correcting codes, such as LDPC codes, have been adapted for use in QKD with great success. Good codes have been obtained which perform very close to the Shannon limit [88, 91] with $\eta_{\mathrm{EC}}(q) < 1.1$, taking full advantage of the advancements in the broader field. LDPC codes, discussed in detail in the next section, also turn out to be well suited for satellite links because of their low communication overhead and the ability to perform most of the processing at one side of the channel.

Author Contributions

I designed and implemented our LDPC-based error correction protocol based on published research by Elkouss [88, 91, 92], Martinez-Mateo [93–95], Pearson et al. [71, 96] and many other unpublished LDPC resources publicly available on the Internet. I modified Hu et al.'s open-source Progressive-Edge Growth (PEG) software [97, 98] and employed it for parity-check matrix construction. I significantly enhanced and optimized the C# decoder developed by Chris Erven [99], which was in turn based on the Matlab code provided by Philip Chan in his Masters thesis [100].

3.6.2 Low-Density Parity-Check Codes

Low-density parity-check (LDPC) codes were first proposed fifty years ago by Gallager [101] but did not gain traction due to the limited hardware capabilities at the time. It was not until MacKay and Neal [70] rediscovered them in 1997 that LDPC codes became more popular. They have recently been widely adopted for classical communication and optimized for a variety of classical channels. This increased interest was due to MacKay and Neal [70] showing that LDPC codes are in fact capacity-approaching, i.e. they can perform close to the Shannon limit, similar to the much more complex turbo codes invented in 1993 by Berrou et al. [102].

In the context of QKD, LDPC codes were first introduced by Pearson et al. [71, 96] at BBN Technologies as part of the DARPA network, where Cascade proved to be very impractical over long distances with high key rates. Other QKD experiments [10, 72] and commercial implementations [24, 25, 103] have followed suit. Recent work by Elkouss [88, 91, 92] and Martinez-Mateo [93–95] has dramatically improved the codes' achievable efficiency, reported in [88] to be as low as $\eta_{\mathrm{EC}} \approx 1.02$ (only 2% from the Shannon limit) for some QBER values.

As implied by their name, LDPC codes are linear[2] block codes which are fully specified by a low-density, binary $M \times N$ parity-check matrix H with entries

$$H_{ij} \in \mathbb{F}_2 = \{0, 1\}. \tag{3.7}$$

The sparse parity-check matrix H is used to compute an M-bit *syndrome*

$$\mathbf{s} = H\mathbf{x} \pmod 2 \tag{3.8}$$

where $\mathbf{x}$ is Bob's N-bit sifted key (column) vector. Each syndrome bit s_i contains parity information from the corresponding parity-check equation (see section 3.6.4) defined by the i^{th} row of H.

The LDPC error correction protocol employed in our high-loss QKD experiment is comprised of the following steps:

1. Alice at the "ground station" prepares a parity-check matrix H as discussed in section 3.6.3. She then transmits H in a compact form to Bob over the classical channel.

2. For each block in his sifted key, Bob on the "satellite" efficiently computes an array of syndromes (section 3.6.4) and streams it down to Alice.

[2] In classical coding theory, a binary block code $\mathcal{C}(n, k)$ with block length n, dimension k and size $|\mathcal{C}| = 2^k$ is linear if $\mathcal{C}$ is a k-dimensional subspace of the n-dimensional binary vector space $(\mathbb{F}_2)^n$, i.e. $\mathcal{C}(n, k) \subseteq (\mathbb{F}_2)^n$.

3. Alice attempts to reconcile her sifted key assuming that Bob's sifted key is "correct". For each N-bit block of sifted key, Alice's goal is to resolve Bob's key vector $\mathbf{x}$, based on her key vector $\mathbf{y}$, Bob's syndrome $\mathbf{s}$, the parity-check matrix H, and the QBER estimate from the coincidence analysis step (section 3.4). To accomplish this task, Alice employs a procedure known as *belief propagation*—an iterative message passing decoding algorithm which is described in section 3.6.3.

All of the above steps are integrated into the QKD software (section 3.3), however, due to the limited capabilities of the computer used at Alice, the error correction procedure is done offline (i.e. not in real time). Furthermore, it is interesting to note that, in the context of QKD, we assume that all communication performed over the classical channel is error-free. Hence, the classical channel implementation (e.g. an RF link) must guarantee that this condition is met by possibly employing standard classical error correcting codes.

3.6.3 Ground-Side Processing

Following our architectural principle of offloading hard computations to the ground station server, we have assigned Alice the tasks of constructing an LDPC matrix and syndrome decoding. Those are both computationally expensive tasks which are fortunately quite suitable for parallelization and hardware acceleration.

Parity-Check Matrix Design

As mentioned, an LDPC code is described by a parity-check matrix H. As depicted in figure 3.8, H can be conveniently visualized with a Tanner [104] graph, an undirected bipartite graph in which each edge connects a *key* node[3] k_j (representing a bit in the sifted key) with a *check* node c_i representing a set of bits from the key used in the corresponding parity-check equation. For example, assume Bob has a 6-bit sifted key vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \tag{3.9}$$

and H is as given in figure 3.8. Then, the parity-check equations producing the 4-bit syndrome $\mathbf{s}$ are

[3]The *key* nodes are also commonly referred to as "message", "bit" or "symbol" nodes in the literature.

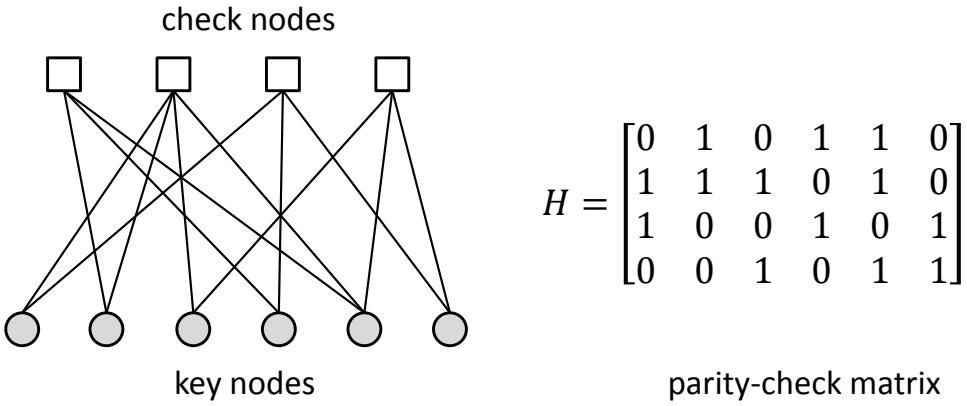

Figure 3.8: Tanner graph representation of a parity-check matrix H. A *key* node (gray circle) represents a bit in the sifted key block vector. A *check* node (square) is connected to all key nodes involved in a given parity-check equation (corresponding to a row of H). The number of ones in H is equal to the number of edges in the graph.

$$s_1 = x_2 \oplus x_4 \oplus x_5 \tag{3.10}$$

$$s_2 = x_1 \oplus x_2 \oplus x_3 \oplus x_5 \tag{3.11}$$

$$s_3 = x_1 \oplus x_4 \oplus x_6 \tag{3.12}$$

$$s_4 = x_3 \oplus x_5 \oplus x_6 \tag{3.13}$$

where $\oplus$ signifies addition modulo 2 (equivalent to the binary XOR operation). Thus, we can easily observe that the number of edges in the Tanner graph corresponds to the number of non-zero entries in H.

The degrees (i.e. the number of incident edges) of the vertices in the graph determine the *degree distribution* of the code. If the degrees of all key and check nodes are respectively constant, i.e. $\deg(k_j) = d_k$ and $\deg(c_i) = d_c$ for all $i \in \{1, \ldots, M\},\ j \in \{1, \ldots, N\}$, then the code is (d_k, d_c)–regular, and hence there are d_k ones in each column and d_c ones in each row of H. The parameters d_k and d_c are commonly referred to as the *column weight* and the *row weight* of H respectively. Randomly generated, regular LDPC codes were the first studied [70], however, *irregular* ones with carefully selected non-constant degree distributions have been shown to have better performance [105].

A family of irregular LDPC codes is generally specified with two generating polynomi-

als [91, 95, 105]:

$$\lambda(x) = \sum_{j=2}^{d_k^{\max}} \lambda_j x^{j-1} \ , \quad \sum_{j=2}^{d_k^{\max}} \lambda_j = 1 \tag{3.14}$$

$$\rho(x) = \sum_{i=2}^{d_c^{\max}} \rho_i x^{i-1} \ , \quad \sum_{i=2}^{d_c^{\max}} \rho_i = 1 \tag{3.15}$$

where $0 \leq \lambda_j \leq 1$ is the fraction of edges (in the Tanner graph) incident to key nodes of degree j, and $0 \leq \rho_i \leq 1$ is the corresponding fraction for check nodes. Also, $d_k^{\max} = \max_j\{\deg(k_j)\}$ and $d_c^{\max} = \max_i\{\deg(c_i)\}$ are respectively the maximum key and check node degrees.

In the context of QKD, the rate of the code is equal to the ratio of unrevealed key bits to the total number of bits per block. There are M rows in H, which correspond to M parity-check equations. Hence, a total of M sifted key bits are revealed per block and the code rate is given by:

$$R = \frac{N - M}{N} \tag{3.16}$$

The coding rate of a family of LDPC codes can also be expressed in terms of the corresponding degree distributions [91, 95]:

$$R = 1 - \frac{\sum_{i=2}^{d_c^{\max}} \rho_i / i}{\sum_{j=2}^{d_k^{\max}} \lambda_j / j} \tag{3.17}$$

More importantly, for the case of our channel model (figure 3.7), the rate is closely related to the code's efficiency and the QBER:

$$\eta_{\mathrm{EC}}(q) = \frac{1 - R}{\mathcal{H}(q)} \tag{3.18}$$

where $\mathcal{H}(q)$ is the binary Shannon entropy

$$\mathcal{H}(q) = -q \log_2 q - (1 - q) \log_2 (1 - q) \tag{3.19}$$

Good LDPC code families for the BSC are published in [91, 95, 105], where the authors employ genetic algorithms called *density evolution* [106] and *differential evolution* to search for optimal degree distributions. Table 3.2 presents a subset of these distributions (with slight modifications) at various code rates which are used in our experiment. According to table 3.2 and equation (3.18), codes can be selected based on how closely their rate corresponds to the QBER estimate for a predefined efficiency value.

Rate	$\lambda(x)$
0.40	$0.18175x + 0.14733x^2 + 0.05443x^3 + 0.07073x^4 + 0.06869x^6 + 0.13514x^8 + 0.15958x^{34} + 0.18235x^{39}$
0.50	$0.15967x + 0.12187x^2 + 0.11261x^3 + 0.19087x^4 + 0.07706x^9 + 0.33792x^{24}$
0.55	$0.16880x + 0.20994x^2 + 0.18095x^5 + 0.03846x^{14} + 0.02635x^{15} + 0.23454x^{17} + 0.05815x^{18} + 0.08280x^{30}$
0.60	$0.11653x + 0.12565x^2 + 0.10851x^3 + 0.05342x^4 + 0.07272x^6 + 0.03479x^7 + 0.07299x^8 + 0.07526x^{17} + 0.11712x^{31} + 0.22301x^{44}$
0.65	$0.10451x + 0.15652x^2 + 0.08057x^3 + 0.00056x^4 + 0.12151x^8 + 0.10485x^{12} + 0.10719x^{14} + 0.00771x^{20} + 0.31656x^{50}$
0.70	$0.09169x + 0.17141x^2 + 0.06839x^3 + 0.12052x^4 + 0.18747x^{10} + 0.20828x^{27} + 0.15224x^{29}$
0.80	$0.09420x + 0.18088x^2 + 0.11972x^5 + 0.08550x^6 + 0.09816x^7 + 0.07194x^{16} + 0.34960x^{25}$
0.90	$0.07689x + 0.28096x^2 + 0.08933x^4 + 0.19620x^6 + 0.30631x^7 + 0.05031x^{11}$

Table 3.2: Generating polynomials corresponding to optimized degree distributions for different code rates, slightly modified from [91, 95].

A major challenge for satellite-based QKD is the limited length of the sifted key produced over a single satellite pass [18]. Finite-size effects [19] aside, this limitation restricts the efficiency of LDPC codes since most optimization schemes assume infinite block lengths. As seen in chapter 5 and [18], the achievable sifted key length over a single satellite pass is around $20,000$ to $100,000$ bits, while good efficiencies have been reported for block sizes in the order of 10^6 [91]. Large block sizes are beneficial because the *sum-product* decoding algorithm (discussed later on) operates optimally over Tanner graphs without any cycles; as the block length is finite, this condition is impossible to satisfy. Intuitively, the length of the smallest cycle in the Tanner graph, i.e. the *girth* of the graph, should be as large as possible [95].

Progressive Edge-Growth Matrix Construction

For small block sizes, randomly constructed parity-check matrices have a much greater chance of containing many small cycles in their Tanner graphs. To solve this problem, Hu et al. [97, 107] proposed an efficient construction procedure—the progressive edge-growth (PEG) algorithm—which takes a combinatorial approach to produce large-girth Tanner graphs. As its name suggests, the PEG algorithm progressively (i.e. edge-by-edge) creates edges between key and check nodes [97]. PEG is a greedy algorithm in the sense that each step aims to maximize the local girth of the subgraph containing the currently considered key node. However, in [97], Hu et al. also outline a non-greedy version of the algorithm in which a desired target girth value for the resulting Tanner graph can be specified.

The PEG algorithm takes as inputs the block size n (i.e. the number of key nodes) and the key node degree distribution $\lambda(x)$ given by equation (3.14), such as the distributions found in table 3.2, to produce a key-node-degree sequence. The algorithm then repeats the following two steps until the desired number of edges have been created [95, 97]:

1. *Subgraph Expansion*: Starting from a key node k_j, the subgraph is expanded such that the key node is connected to the most distant check node. Check nodes producing short cycles are detected and avoided, and a set $\overline{\mathcal{N}}^l_{k_j}$ of candidate check nodes producing long cycles is created. $\mathcal{N}^l_{k_j}$ is the set of check nodes reachable from the key node k_j after subgraph expansion up to depth l.

2. *Edge-Selection*: Update $\overline{\mathcal{N}}^l_{k_j}$ according to the key-node-degree sequence and by selecting candidate check nodes with the lowest degree.

An improved version of this algorithm is presented in [94] where the check node degree distribution $\rho(x)$ given by equation (3.14) is also taken into account.

Our QKD software incorporates a modified version of Hu et al.'s open-source PEG parity-check matrix construction software [97, 98] employing the optimal degree distribution profiles shown in table 3.2. The PEG construction has significantly improved the performance of our LDPC codes with moderate block lengths around $30,000$ to $50,000$ bits. However, for block sizes under $30,000$, a major difficulty becomes to produce matrices corresponding to Tanner graphs without cycles of length four; yet, still satisfying the desired degree distribution profiles as given in table 3.2. Intuitively, as the size of the parity-check matrix is increased, the degree distribution profile constraints become easier to satisfy, because the required number of edges grows linearly with the block size, while the total number of entries in H grows quadratically.

Belief Propagation Decoding

Recall that according to the protocol described in section 3.6.2, after Alice has produced a PEG-generated parity check matrix, H, and Bob has computed and sent down his syndrome, $\mathbf{s}$, Alice needs to reconcile her key, $\mathbf{y}$, so that it matches Bob's key, $\mathbf{x}$. The reconciliation is accomplished with a syndrome decoding algorithm which takes H, $\mathbf{s}$ and $\mathbf{y}$ as inputs and, when successful, produces an estimate, $\hat{\mathbf{x}}$, of Bob's key.

The most straightforward way to proceed is to use maximum-likelihood decoding, where the decoder aims to maximize $\Pr(X = \hat{\mathbf{x}} | Y = \mathbf{y})$ and hence the probability that $\hat{\mathbf{x}} = \mathbf{x}$ (see figure 3.7). Unfortunately, given our channel model, there is no known efficient (i.e. polynomial-time) algorithm to accomplish this task. In fact, general maximum-likelihood decoding is an NP-complete problem for the BSC [95, 108].

Alternatively, Gallager [101] suggests the use of message-passing decoding algorithms which have been shown to work efficiently when the codes have sparse parity-check matrices. Message-passing algorithms are iterative algorithms in which messages are propagated between key and check nodes (and vice versa) along the edges of the Tanner graph. When the messages contain probabilities (or beliefs), the technique is called *belief propagation* [95]. For LDPC decoding, the most common belief propagation implementation is known as the *sum-product* algorithm, described in detail in [70]. In the context of QKD, Pearson [71] proposed a small change to this algorithm in order to impose additional parity constraints. A review of the sum-product decoding algorithm for QKD is presented below following notation similar to [70, 100] and [95].

Define the following index sets:

$$\mathcal{N}_i = \{j \mid H(i,j) = 1\}, \quad i \in \{1, \ldots, M\} \tag{3.20}$$

$$\mathcal{M}_j = \{i \mid H(i,j) = 1\}, \quad j \in \{1, \ldots, N\} \tag{3.21}$$

Hence, $\mathcal{N}_i$ indicates which bits from Bob's sifted key vector $\mathbf{x}$ participate in the i^{th} parity-check equation, as determined by the non-zero entries in the i^{th} row of H. Conversely, $\mathcal{M}_j$ specifies which parity-check equations incorporate the key bit x_j, as determined by the non-zero entries in the j^{th} column of H.

Let $\mathbf{p}^0$ and $\mathbf{p}^1$ be initial (a priori) probability vectors such that

$$p_j^t = \Pr(x_j = t|y_j), \quad t \in \{0,1\} \tag{3.22}$$

and observe that

$$p_j^1 = 1 - p_j^0 \tag{3.23}$$

Similarly, $\mathbf{q}^0$ and $\mathbf{q}^1$ are the final (pseudo-posterior) probability vectors produced by the decoding algorithm.

Furthermore, Q^0 and Q^1 are two matrices storing the messages going from key to check nodes. Each message Q_{ij}^t ($t \in \{0,1\}$) contains the probability that $x_j = t$ based on the information from the parity-check equations in $\mathcal{M}_j \backslash \{i\}$. Correspondingly, the matrices R^0 and R^1 store the messages passed in the opposite direction—from check nodes to key nodes. Each message R_{ij}^t ($t \in \{0,1\}$) contains the probability that the i^{th} parity-check equation is satisfied given that $x_j = t$ is fixed and the probability distribution

$$\{Q_{ij'}^t \mid j' \in \mathcal{N}_i \backslash \{j\}\} \tag{3.24}$$

for the remaining key bits is separable [70].

The sum-product algorithm has an initialization stage followed by two sequential message-passing steps, which are executed iteratively until successful decoding or until a predefined number of iterations has been reached.

1. *Initialization*: Suppose the initial QBER estimate is q, then $\mathbf{p}^0$ and $\mathbf{p}^1$ are initialized as

$$\forall t \in \{0,1\}, \quad \forall j \in \{1,\ldots,N\}, \quad p_j^t = \begin{cases} 1-q & \text{if } y_j = t \\ q & \text{if } y_j \neq t \end{cases} \tag{3.25}$$

   Also, the initial messages from key to check nodes are set to

$$\forall t \in \{0,1\}, \quad \forall j \in \{1,\ldots,N\}, \quad \forall i \in \mathcal{M}_j, \quad Q_{ij}^t = p_j^t \tag{3.26}$$

2. *Horizontal step*: The decoder calculates the probability messages going from check to key nodes. Let

$$Q^{\delta} = Q^{0} - Q^{1} \tag{3.27}$$

then the following probabilities are computed:

$$\forall i \in \{1, \ldots, M\}, \quad \forall j \in \mathcal{N}_i, \quad R_{ij}^{\delta} = (-1)^{s_i} \prod_{j' \in \mathcal{N}_i \setminus \{j\}} Q_{ij'}^{\delta} \tag{3.28}$$

$$\forall i \in \{1, \ldots, M\}, \quad \forall j \in \mathcal{N}_i, \quad R_{ij}^{0} = \frac{1 + R_{ij}^{\delta}}{2} \tag{3.29}$$

$$\forall i \in \{1, \ldots, M\}, \quad \forall j \in \mathcal{N}_i, \quad R_{ij}^{1} = \frac{1 - R_{ij}^{\delta}}{2} \tag{3.30}$$

where R^{δ} is a sparse matrix storing intermediate results, and s_i is the i^{th} bit in Bob's syndrome, $\mathbf{s}$. In equation (3.28) the sign of R_{ij}^{δ} is changed whenever $s_i = 1$. This is the modification for QKD suggested by Pearson [71].

3. *Vertical step*: The decoder calculates the probability messages going from key to check nodes as follows:

$$\forall t \in \{0, 1\}, \quad \forall j \in \{1, \ldots, N\}, \quad \forall i \in \mathcal{M}_j, \quad Q_{ij}^{t} = \alpha_{ij}\, p_j^{t} \prod_{i' \in \mathcal{M}_j \setminus \{i\}} R_{i'j}^{t} \tag{3.31}$$

Next, the pseudo-posterior probability vectors are computed:

$$\forall t \in \{0, 1\}, \quad \forall j \in \{1, \ldots, N\}, \quad q_j^{t} = \alpha_j\, p_j^{t} \prod_{i \in \mathcal{M}_j} R_{ij}^{t} \tag{3.32}$$

where α_{ij} and α_j are normalization factors which are added to ensure that

$$\forall j \in \{1, \ldots, N\}, \quad \forall i \in \mathcal{M}_j, \quad Q_{ij}^{0} + Q_{ij}^{1} = 1 \tag{3.33}$$

and

$$\forall j \in \{1, \ldots, N\}, \quad q_j^{0} + q_j^{1} = 1 \tag{3.34}$$

4. *Termination Condition*: Based on the pseudo-posterior probabilities obtained in equation (3.32), the estimate, $\hat{\mathbf{x}}$, of Bob's key is subsequently updated:

$$\forall j \in \{1, \ldots, N\}, \quad \hat{x}_j = \begin{cases} 1 & \text{if } q_j^{1} > \frac{1}{2} \\ 0 & \text{if } q_j^{1} \leq \frac{1}{2} \end{cases} \tag{3.35}$$

The algorithm terminates with SUCCESS if Alice's syndrome matches the syndrome obtained from the estimate $\hat{\mathbf{x}}$, i.e. when

$$H\mathbf{y} = H\hat{\mathbf{x}} \tag{3.36}$$

If the maximum number of iterations has been reached and the above condition is not satisfied, the algorithm terminates with FAILURE. Otherwise, a new iteration begins at the horizontal step.

Note that it is not necessary to store H itself as the index sets $\mathcal{N}_i$ and $\mathcal{M}_j$ already contain the full information given by the parity-check matrix. The six matrices $Q^0, Q^1, Q^\delta, R^0, R^1, R^\delta$ are sparse (with non-zero entries in the same locations as H), and $\mathcal{N}_i, \mathcal{M}_j$ indicate the positions of the non-zero entries across all of them.

In our QKD software, the sum-product decoder can be run in parallel on multiple blocks of sifted key, in a way similar to the coincidence algorithm (section 3.4.1). The computational complexity per block scales linearly with the block length, because H is sparse and the row weights are kept under a small constant value (usually less than 50).

3.6.4 Satellite-Side Processing

On the satellite, the required computation is minimal—as desired. All Bob needs to do is compute his syndromes after obtaining H from Alice on the ground. This operation can be accomplished with a simple matrix multiplication:

$$\mathbf{s} = H\mathbf{x} \pmod 2 \tag{3.37}$$

However, this computation does not take advantage of the sparseness of H.

There is a more efficient way to calculate the syndrome if H is stored in the adjacency list format. The procedure is presented in algorithm 3.3. Since the row weights of H never exceed a small constant value, the innermost **foreach** loop in algorithm 3.3 executes a bounded number of iterations. Thus, the computational complexity per block scales linearly with the block size.

Another benefit of this approach is that Alice does not need to transmit the full matrix H to the satellite. Following the notation from the previous section, Alice effectively sends Bob only the N index sets $\mathcal{M}_j$ containing the non-zero entries in each column j of H.

Algorithm 3.3: Efficient syndrome computation

Data: A binary parity-check matrix H represented in the adjacency list format, a binary vector $\mathbf{x}$ storing a block of sifted key

Result: A binary vector containing the syndrome $\mathbf{s} = H\mathbf{x}$

```
/* Initialize s to a 0-filled vector                                  */
```
1 $\mathbf{s} = \mathbf{0}$

```
/* adjList[j] holds a list of indices i for which H[i, j] = 1         */
```
2 **for** *j = 1 to x.length* **do**
3 **if** *x[j] == 1* **then**
4 **foreach** *index i in adjList[j]* **do**
5 $s[i] = s[i]$ XOR 1
6 **end**
7 **end**
8 **end**
9 **return** s

3.7 Privacy Amplification

If the information reconciliation step described in section 3.6 terminates successfully, Alice and Bob share the error-corrected key

$$\mathbf{k}_{\mathrm{EC}} = \hat{\mathbf{x}} = \mathbf{x} \tag{3.38}$$

However, $\mathbf{k}_{\mathrm{EC}}$ is only partially secure, since some information might have been leaked out to the eavesdropper, Eve, either during the quantum signal exchange (as evidenced by the QBER which we must attribute to Eve) or during error correction, where we assume that all parity information is known to Eve.

A procedure known as privacy amplification [109–111] is employed to reduce Eve's partial information about $\mathbf{k}_{\mathrm{EC}}$. As depicted in figure 3.9, privacy amplification consists of applying a compression function f to the partially-secure, error-corrected key $\mathbf{k}_{\mathrm{EC}}$ to produce a provably secure key $\mathbf{k}_{\mathrm{F}}$ of length $L < N$, where N is the key block size. The length L of the final secure key is the subject of study of QKD security proofs [19, 64, 112, 113] as discussed in section 1.2. Due to finite-size effects, N needs to be kept above a certain value (usually on the order of 10^5) and that has to be taken into account when selecting a hash function.

Privacy amplification is unfortunately a symmetric procedure which needs to be performed by both parties. Thus, a great amount of work has gone into reducing its complexity.

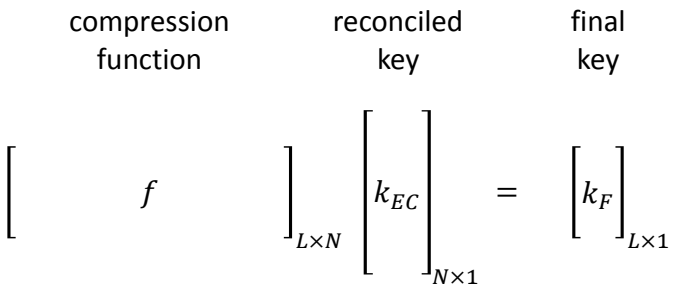

Figure 3.9: Privacy amplification in general. After the error correction step, Alice and Bob randomly choose a common compression function f, which is applied to the reconciled key $\mathbf{k}_{\mathrm{EC}}$ to produce the final secure key $\mathbf{k}_{\mathrm{F}}$ of length $L < N$.

3.7.1 Two-Universal Hash Functions

There are several considerations when selecting the hash function

$$f : \{0,1\}^N \to \{0,1\}^L \tag{3.39}$$

Firstly, f needs to be chosen such that Eve's information about $\mathbf{k}_{\mathrm{F}}$ is reduced below some infinitesimal value. In Shor-Preskill's[64, 112] and Scarani-Renner's[19, 113] formalisms, the key can be secured by a special family of functions called *two-universal hash functions* [1, 114].

Definition 1. *A family of functions*

$$\mathcal{F} = \{f_r \mid r \in \mathbb{N}, f_r : A \to B\}$$

is ***two-universal*** *if*

$$Pr\{f_r(x) = f_r(\hat{x})\} \leq \frac{1}{|B|}, \quad \forall\, x \neq \hat{x} \in A$$

Two-universal hash functions are used for privacy amplification in the following way:

1. After the information reconciliation step, Alice and Bob share the error-corrected key $\mathbf{k}_{\mathrm{EC}}$ and Eve has an estimate $\hat{\mathbf{k}}_{\mathrm{EC}}$.

2. Alice and Bob choose f from the two-universal set and communicate it publicly. They then share a shorter final key $\mathbf{k}_{\mathrm{F}} = f(\mathbf{k}_{\mathrm{EC}})$.

3. Eve's estimate $\hat{\mathbf{k}}_{\mathrm{F}} = f(\hat{\mathbf{k}}_{\mathrm{EC}})$, has a very low probability of coinciding with $\mathbf{k}_{\mathrm{F}}$, since by two-universality (definition 1),

$$\Pr(\mathbf{k}_{\mathrm{F}} = \hat{\mathbf{k}}_{\mathrm{F}}) \leq \frac{1}{2^L} \tag{3.40}$$

Thus, Eve has the same chances of obtaining the key as if she chose randomly from the set $\{0,1\}^L$ of all potential final keys [1].

Note that f must be chosen randomly with uniform probability from the set of two-universal hash functions *after* the measurement, so that Eve cannot adapt her eavesdropping strategy.

3.7.2 LFSR Implementation

When QKD post-processing is done with limited resources, major considerations for the choice of hash function are the computational complexity and the amount of classical communication required. In our QKD implementation, the privacy amplification procedure roughly follows the methodology outlined in [114], however, we made some modifications to their model (which had some inaccuracies) and developed a different matrix multiplication procedure presented in algorithm 3.4. In brief, we employ the Toeplitz matrix [115] construction implemented efficiently with a linear feedback shift register (LFSR).

Definition 2. *A diagonal-constant matrix or* ***Toeplitz matrix*** *is a matrix which has constant descending left-to-right diagonals. An $L \times N$ Toeplitz matrix can be written in the following form:*

$$T_r = \begin{bmatrix} r_L & r_{L+1} & \cdots & \cdots & & \cdots & \cdots & r_{N+L-2} & r_{N+L-1} \\ r_{L-1} & r_L & \ddots & & & & \cdots & r_{N+L-3} & r_{N+L-2} \\ \vdots & r_{L-1} & \ddots & \ddots & & & & \vdots & \vdots \\ r_2 & \vdots & \ddots & r_L & r_{L+1} & & & & r_{N-2} \\ r_1 & r_2 & \cdots & r_{L-1} & r_L & r_{L+1} & \cdots & \cdots & r_{N-1} \end{bmatrix}_{L \times N}$$

In [73], Krawczyk shows that Toeplitz matrices are in fact two-universal hash functions. Note that the Toeplitz matrix T_r above is completely defined by the $(N+L-1)$-bit vector

$$\mathbf{r} = (r_1, r_2, \ldots, r_{N+L-1}) \tag{3.41}$$

so there is no need to store T_r or to transmit the entire matrix over the classical channel. Thus, these matrices are very suitable for privacy amplification.

A simplification to the construction in definition 2 is proposed by Hayashi et al. [116, 117]. It has been shown that a matrix of the following form

$$H_r = (I_L | T_r) =$$

$$\begin{bmatrix}
1 & 0 & \dots & \dots & 0 & r_L & r_{L+1} & \dots & \dots & & \dots & \dots & r_{N-2} & r_{N-1} \\
0 & 1 & \ddots & & \vdots & r_{L-1} & r_L & \ddots & & & & \dots & r_{N-3} & r_{N-2} \\
\vdots & 0 & \ddots & 0 & \vdots & \vdots & r_{L-1} & \ddots & \ddots & & & & \vdots & \vdots \\
\vdots & & \ddots & 1 & 0 & r_2 & \vdots & \ddots & r_L & r_{L+1} & & & & r_{N-L-1} \\
0 & \dots & \dots & 0 & 1 & r_1 & r_2 & \dots & r_{L-1} & r_L & r_{L+1} & \dots & \dots & r_{N-L}
\end{bmatrix} \tag{3.42}$$

i.e. a concatenation of an identity matrix I_L and a Toeplitz matrix T_r, is also two-universal, however, it requires only $N-1$ bits to define.

Using the streamlined Toeplitz construction from equation (3.42), we have implemented the following privacy amplification protocol:

1. After the information reconciliation step, Alice generates a random binary string

$$\mathbf{r} = (r_1, r_2, \dots, r_{N-1}) \tag{3.43}$$

   of length $N-1$.

2. Alice transmits $\mathbf{r}$ over the classical channel to Bob.

3. Alice and Bob use $\mathbf{r}$ and an LFSR to compute the final secure key

$$\mathbf{k}_{\mathrm{F}} = H_r \, \mathbf{k}_{\mathrm{EC}} \tag{3.44}$$

   as described in figure 3.10 and algorithm 3.4.

Note that step 3 involves the same amount of computation on the satellite as on the ground, and hence, it could potentially create a bottleneck on the satellite. Fortunately, the matrix multiplication in equation (3.44) can be implemented with efficient bitwise operations and an LFSR. Figure 3.10 depicts our implementation graphically. The full procedure is given in algorithm 3.4.

Algorithm 3.4: LFSR-based privacy amplification

Data: Error-corrected key vector $\mathbf{k}_{\mathrm{EC}}$, $(N-1)$-bit random binary string $\mathbf{r}$. Both stored in 32-bit unsigned integer arrays. The final key length L.

Result: Final key vector $\mathbf{k}_{\mathrm{F}} = H_r\ \mathbf{k}_{\mathrm{EC}}$

```
1  temp = 0, k_F = 0
2  numLFSRUnits = (N-L)/32
3  for i = 1 to numLFSRUnits do
4  |  lfsr[i] = r[L/32 + i]
5  |  temp = temp + LookupBitCount ( lfsr[i] AND k_EC[L/32 + i] )
6  end
7  if ( 1 AND k_EC[1] ) ≠ 0 then
8  |  temp = temp + 1
9  end
10 if ( temp MOD 2 ) ≠ 0 then
11 |  k_F[1] = 1
12 end
13 for i = 2 to numLFSRUnits do
14 |  temp = 0
15 |  sbit = GetNextInputBit (r)
16 |  for j = 1 to numLFSRUnits do
17 |  |  fbit = 0
18 |  |  if ( lfsr[j] AND 0×80000000 ) ≠ 0 then
19 |  |  |  fbit = 1
20 |  |  end
21 |  |  lfsr[j] = lfsr[j] LSHIFT 1
22 |  |  lfsr[j] = lfsr[j] OR sbit
23 |  |  sbit = fbit
24 |  |  temp = temp + LookupBitCount ( lfsr[j] AND k_EC[L/32 + j] )
25 |  end
26 |  if ( (1 LSHIFT (i MOD 32)) AND k_EC[i] ) ≠ 0 then
27 |  |  temp = temp + 1
28 |  end
29 |  if ( temp MOD 2 ) ≠ 0 then
30 |  |  k_F[i] = k_F[i] OR (1 LSHIFT (i MOD 32) )
31 |  end
32 end
33 return k_F
```

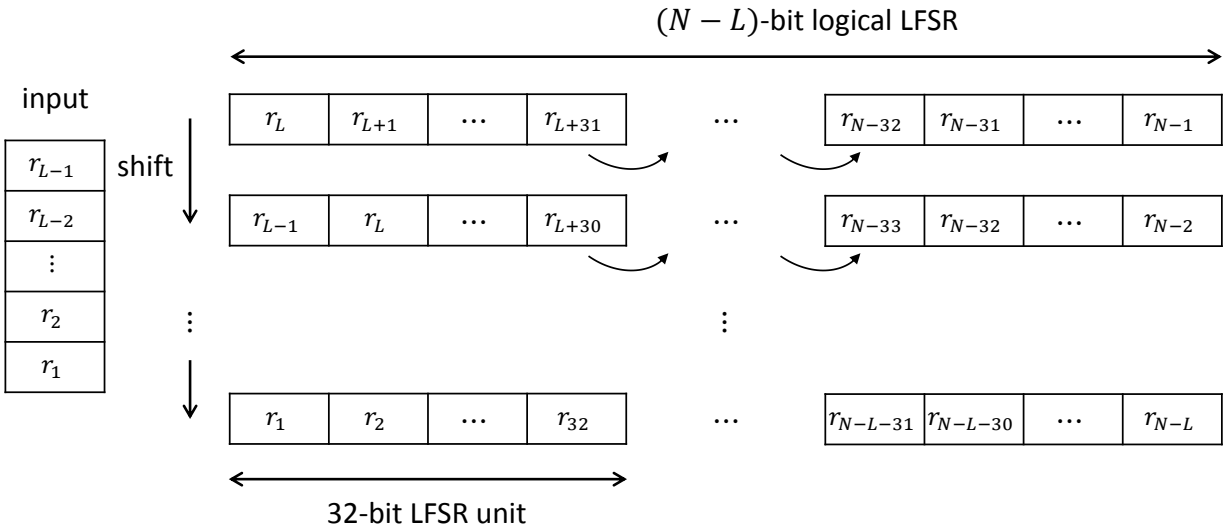

Figure 3.10: LFSR-based implementation of Toeplitz matrix multiplication.

Definition 3. *A **linear feedback shift register** (LFSR) is a shift register in which the input bit is a linear function of the register's previous state.*

The main idea behind our memory-constrained privacy-amplification implementation is to never store full matrices. The identity portion of each row of $H_r = (I_L|T_r)$ takes up no space and can be accounted for with a simple AND operation (see algorithm 3.4). We represent the Toeplitz matrix T_r, which is embedded in H_r from equation (3.42), as an $(N-L)$-bit logical LFSR. Initially, the LFSR contains the bit vector

$$\mathbf{r}_{\text{init}} = (r_L, r_{L+1}, \ldots, r_{N-1}) \tag{3.45}$$

and the remaining bits from $\mathbf{r}$ are used as input for the LFSR

$$\mathbf{r}_{\text{input}} = (r_{N-L-1}, r_{N-L-2}, \ldots, r_1) \tag{3.46}$$

Since $(N-L)$ bits cannot fit in a single register, the logical LFSR is broken up into multiple 32-bit LFSR blocks as illustrated in figure 3.10. Each block is designed to fit inside a register on a processing unit. The register size of 32 is chosen because that number is widely supported across multiple platforms including our low-power ARM test board. 64-bit platforms are also available, and with single instruction, multiple data (SIMD) extensions, the register can be as large as 128 bits. The only drawback to using larger registers is that the granularity is reduced, as the values of N and L need to be multiples of the register size.

After initialization, algorithm 3.4 computes each row of the multiplication $S_r \, \mathbf{k}_{\mathrm{EC}}$ by performing a bitwise AND of each LFSR unit and the corresponding section of $\mathbf{k}_{\mathrm{EC}}$, while keeping track of the bit count (implemented with an efficient table lookup). Each row of S_r produces a bit in the final key. For the next row, all LFSR units are shifted. The first unit gets an input bit from $\mathbf{r}_{\mathrm{input}}$ and all other units get a feedback bit from their leftmost neighbor as shown in figure 3.10. The process continues until no more bits are left in $\mathbf{r}_{\mathrm{input}}$. Nevertheless, this is still an $\Theta(N^2)$ algorithm. However, the constant is very small, on the order of $\frac{1}{4000}$.

Tsurumaru [114] suggests a further optimization which can be applied, assuming that both parties are able to quickly compute fast Fourier transforms (FFTs). It consists of embedding the Toeplitz matrix T_r from definition 2 in a circulant matrix C_r and breaking up the multiplication as described in [118]:

$$\mathbf{k}_{\mathrm{F}}^{'} = C_r \, \mathbf{k}_{\mathrm{EC}}^{'} \tag{3.47}$$

$$= F^{-1} F \, C_r \, F^{-1} F \, \mathbf{k}_{\mathrm{EC}}^{'} \tag{3.48}$$

$$= F^{-1} \, \mathrm{diag}(F\mathbf{r}) \, F \, \mathbf{k}_{\mathrm{EC}}^{'} \tag{3.49}$$

where $\mathbf{k}_{\mathrm{EC}}^{'}$ is just $\mathbf{k}_{\mathrm{EC}}$ padded with zeros at the bottom to match the dimensions of C_r. Similarly, the desired result of the computation, i.e. $\mathbf{k}_{\mathrm{F}}$, is found in the top L bits of $\mathbf{k}_{\mathrm{F}}^{'}$.

In terms of complexity, this approach is faster as it runs in time $\Theta(N \log_2 N)$. However, it requires access to an FFT engine. FFT can be implemented without hardware acceleration, but it still requires floating-point support, which was something we did not want to assume. Hence, we stuck with the $\Theta(N^2)$ algorithm which works well with smaller block sizes due to the small constant factor (see chapter 4).

Chapter 4

Post-Processing Resource Requirements

Important practical consideration for the implementation of a QKD receiver on a satellite platform are the computational resource requirements of the satellite-side QKD protocol as well as the communication bandwidth needed to accomplish all classical post-processing steps. The software stack implementing the QKD protocol (chapter 3) is designed in such a way that computing resource requirements and network bandwidth usage can be accurately estimated. Estimates of the computing resources required onboard the satellite are discussed in section 4.1. Classical communication requirements are provided in section 4.2

4.1 Satellite-Side Resources

Recall that in the proposed uplink scenario discussed in section 1.3, optical signals are sent from the ground when the satellite orbits over an optical ground station, while classical communication is performed (at a later time) when the satellite orbits over one or more RF ground stations. Hence, the satellite system needs to store all time-tags accumulated during the optical station flyover, and then perform all steps of the QKD protocol during an RF station flyover (when a classical communication link is present). The estimates described in sections 4.1.1 and 4.1.2 assume that each flyover lasts approximately 5 minutes.

4.1.1 Memory Requirements

In terms of memory requirements, Bob must store: time-tags, measurement basis, photon detections (stored as bit values), LDPC matrix, and the privacy amplification Toeplitz matrix [115] which is efficiently implemented with a shift register [73]. Full details of our protocol implementation are provided in chapter 3.

Parameter	Value	Unit
Time-tag precision	78	picosec
Time register width	34	bit
Channel register width	3	bit
Total bits per time-tag	37	bit
Total with byte alignment	40	bit
Duration of measurement	300	sec
Maximum received average time-tag rate	100	kHz
Maximum total memory usage	**150**	**Mbyte**

Table 4.1: Memory requirements for time-tag storage.

Data Storage

The time-tagging hardware (see section 3.1) produces time-tags of size up to 64 bits. However, to save memory (and classical communication traffic) it is possible to reduce that number significantly, at the expense of additional computation steps. One simple scheme is to store the full time-tag only at the beginning of every second of data collection, together with additional information provided by the GPS receiver (which outputs a data packet every second). The memory requirements for this scheme and supporting parameters are provided in table 4.1.

Error Correction

Our one-way error correction scheme is based on low-density parity check (LDPC) codes (see section 3.6). Its advantage is a significant reduction of computation steps on the satellite and a reduction in classical communication. On the satellite side, an M by N sparse parity check binary matrix needs to be stored. It is applied to parts of the key (blocks of size N) to produce syndrome vectors of size M. Thus, the size of the matrix varies based on the choice of block size N and the channel QBER, q. From Shannon's channel coding theorem [83] applied to the binary symmetric channel [82], we can deduce a closed form estimate of the size of the LDPC matrix based on the QBER [88],

$$M = N\eta_{\mathrm{EC}}H(q) \tag{4.1}$$

Here, η_{EC} is the error correction efficiency and H is the binary entropy function.

Parameter	Value	Unit
Sifted key rate	40	kHz
Sifted key buffer	1.5	Mbyte
Block size, N	40000	bit
Error correction efficiency, η_{EC}	1.2	
QBER, q	0.07	
Syndrome length, M	14296	bit
Maximum row weight, W_r	100	
LDPC parity-check matrix	5.7	Mbyte
Total (maximum) memory usage	**7.2**	**Mbyte**

Table 4.2: Memory requirements for error correction. This memory usage can be significantly reduced at the expense of worse error correction efficiency and hence lower final key rate.

We can efficiently store the sparse parity check matrix in the adjacency list format where only the indices of each non-zero element in each row are stored. There are at most $W_r\, M$ such indices, and each index is stored in at most 32 bits. Sample numbers are shown in table 4.2. Note that the block size N can be made significantly smaller at the expense of possibly increasing the value of η_{EC} (i.e. by having worse error correction efficiency).

Privacy Amplification

In this final stage of the QKD protocol, Bob on the satellite receives a random binary string of length equal to the length N of the sifted key. Then he uses an LFSR to implement a Toeplitz matrix (as discussed in section 3.7) and applies it to his key. Due to the efficient matrix implementation, at most N bits need to be stored per block. Thus, the total memory usage based on the parameters indicated in table 4.2 would be 40 kbit.

Total Memory Requirements

Summarizing the estimates above, the total maximum memory required for QKD on the satellite computer is about 157 Mbyte. Therefore, the recommended total system memory should be at least 256 Mbyte. This number will ultimately depend also on the memory requirement of the on-board operating system. Based on the survey in appendix A of state of the art space-grade computing hardware, this memory requirement is well within our current technological capabilities.

Type of operation	Asympt. number of operations	Estimate of constant	Estimated number of operations per second
Comparison	O(n)	5	500,000
memcpy	O(n)	1	100,000
Logical AND/OR	O(n)	2	200,000
Assignment	O(n)	5	500,000
Addition/Subtraction	O(n)	3	300,000
Total operations	O(n)	**16**	**1,600,000**

Table 4.3: Worst-case computational requirements for time-tag processing, where n is the raw key rate.

4.1.2 Computing Requirements

The current state of the satellite-side software (Bob) allows us to obtain estimates of the number of operations required to perform QKD. The estimates are provided in terms of asymptotic analysis of the underlying algorithms. Exact numbers in terms of clock cycles or number of basic instructions per second will be highly dependent on the specific processing architecture used for the onboard computer. Our implementation currently runs both on a standard x86 desktop computer, and on a low-power embedded ARM platform.

Time-Tag Processing

The processing of time-tags is a linear algorithm, so it is only dependent on the total number of detected time-tags. To estimate the number of operations per second, we assume a raw key rate of 100 kHz. Worst-case estimates of the number of operations are given in table 4.3.

Sifting

Sifting is also a linear process. It is dependent on the sifted key rate, which is assumed to be at most 40 kHz. Worst-case estimates are shown in table 4.4.

Error Correction

The error correction algorithm (see section 3.6) on the satellite side consists of multiplying a sparse binary matrix (the LDPC parity check matrix) by a binary vector (a block of

Type of operation	Asympt. number of operations	Estimate of constant	Estimated number of operations per second
Comparison	O(m)	4	160,000
Addition/Subtraction	O(m)	3	120,000
Bit shift	O(m)	1	40,000
Logical AND/OR	O(m)	2	80,000
Assignment	O(m)	5	200,000
Total operations	O(m)	**15**	**600,000**

Table 4.4: Worst-case computational requirements for sifting, where m is the sifted key rate.

Type of operation	Asympt. number of operations	Estimate of constant	Estimated number of operations per second
Comparison	O($W_r\,N$)	100	4,000,000
Logical XOR	O($W_r\,N$)	50	2,000,000
Assignment	O($W_r\,N$)	50	2,000,000
Total operations	O($W_r\,N$)	**200**	**8,000,000**

Table 4.5: Worst-case computational requirements for error correction. $M \times N$ is the size of the LDPC parity check matrix H. $W_r < 100$ is the maximum row weight of H.

sifted key). As discussed in section 4.1.1, the size of the matrix ($M \times N$) depends on the QBER and the desired error correction efficiency according to equation (4.1). Assuming parameter values as indicated in table 4.2 and the sparse-matrix multiplication procedure described in algorithm 3.3, we obtain the computational estimates shown in table 4.5.

Privacy Amplification

Our privacy amplification implementation (section 3.7) is based on a hashing algorithm with Toeplitz matrices [115] which is implemented with a shift register [73, 119]. The hashing algorithm operates on the error corrected key of size N. The resulting final key length L depends on several factors (QBER, error correction efficiency, sifted key length). In practice, $L < \frac{N}{2}$ for our loss levels.

Using parameter values as in table 4.2, we obtain the privacy amplification estimates in table 4.6.

Type of operation	Asympt. number of operations	Estimate of constant	Estimated number of operations per second
Comparison	$O((N-L)L)$	1/32	12,500,000
Addition	$O((N-L)L)$	2/32	25,000,000
Bit shift	$O((N-L)L)$	1/32	12,500,000
Logical AND/OR	$O((N-L)L)$	3/32	37,500,000
Assignment	$O((N-L)L)$	6/32	75,000,000
Total operations	$O((N-L)L)$	**13/32**	**162,500,000**

Table 4.6: Worst-case computational requirements for privacy amplification (assuming a 32-bit computing architecture). N is the error-corrected key length, L is the final key size.

Total Computational Requirements

Based on the above estimates, the overall worst-case computational requirement of the QKD protocol for the satellite receiver is approximately 173 million operations per second. Different architectures/compilers might translate these operations into a different number of basic processor instructions, but experience suggests that for these requirements to be satisfied a clock speed of at least 750 MHz is necessary.

4.2 Classical Communication Requirements

Any QKD system requires a reliable classical channel to execute all post-processing steps. In a satellite-based scenario, classical communication bandwidth is a very important factor. When designing our concept system, we aimed to minimize the uplink (ground to satellite) bandwidth as it is expected to be quite limited for a small satellite. Downlink (satellite to ground) bandwidth is expected to be reasonable.

To evaluate the communication requirements of our post-processing system (chapter 3) at different possible key rates, experimental data is collected (using the apparatus described in chapter 2) for 300 seconds at a receiver detection rate of about 150 kHz. A subset of the data is then used to produce lower key rates in the range we expect to find for satellite-based QKD. Specifically, each one second chunk of data is truncated accordingly to achieve the desired raw key rate. The full QKD protocol is then performed on the resulting data subsets and detailed uplink and downlink bandwidth statistics are collected and recorded.

Parameter	Value	Unit
Time-tag precision	78	picosec
Time register width	34	bit
Channel register width	3	bit
Total bits per time-tag	37	bit
Total with byte alignment	40	bit
Duration of measurement	300	sec
Maximum average received time-tag rate	100	kHz
Time to stream 100k time-tags (2 Mbit/s link)	2	sec
Time to stream 100k time-tags (5 Mbit/s link)	0.8	sec
Time to stream 5 min worth of time-tags (2 Mbit/s link)	**600**	**sec**
Time to stream 5 min worth of time-tags (5 Mbit/s link)	**240**	**sec**

Table 4.7: Communication requirements for the transmission of raw time-tags from the satellite to the ground.

4.2.1 Downlink

The largest portion of the downlink communication requirements come from the transmission of raw tags from the satellite to the ground station. Parameter estimates are provided in table 4.7.

Table 4.8 summarizes downlink communication statistics collected from experimental data at different raw key rates. Our estimates and results show that at least a 5 Mbit/s downlink is required to be able to transmit and process all of the QKD data (collected during the optical station flyover) in a single RF-station flyover.

4.2.2 Uplink

As mentioned, the satellite uplink bandwidth is expected to be limited, so our system aims to minimize the upload rate. Table 4.9 displays uplink communication statistics collected from experimental data at different raw key rates. Our results show that the uplink requirement is indeed much lower than that for downlink. Our recommended uplink bandwidth is 100 kbit/s, however the statistics for raw key rates above 50 kHz are provided for scalability purposes only, as we do not expect the average observable rates to exceed 50 kHz over a single satellite pass.

Raw key rate (Hz)	Time-tag data (byte)	GPS data (byte)	EC data (byte)	Total (byte)	Data rate (kbit/s)
10,000	14,850,000	16,632	65,512	14,932,144	402
20,000	29,700,000	16,632	132,176	29,848,808	804
30,000	44,550,000	16,632	254,764	44,821,396	1,207
40,000	59,400,000	16,632	322,566	59,739,198	1,609
50,000	74,250,000	16,632	389,568	74,656,200	2,011
60,000	89,100,000	16,632	519,086	89,635,718	2,414
70,000	103,950,000	16,632	452,386	104,419,018	2,813
80,000	118,800,000	16,632	519,086	119,335,718	3,214
90,000	133,650,000	16,632	596,898	134,263,530	3,617
100,000	148,500,000	16,632	654,722	149,171,354	4,018

Table 4.8: Downlink (satellite to ground) communication statistics resulting from processing 300 seconds of QKD data.

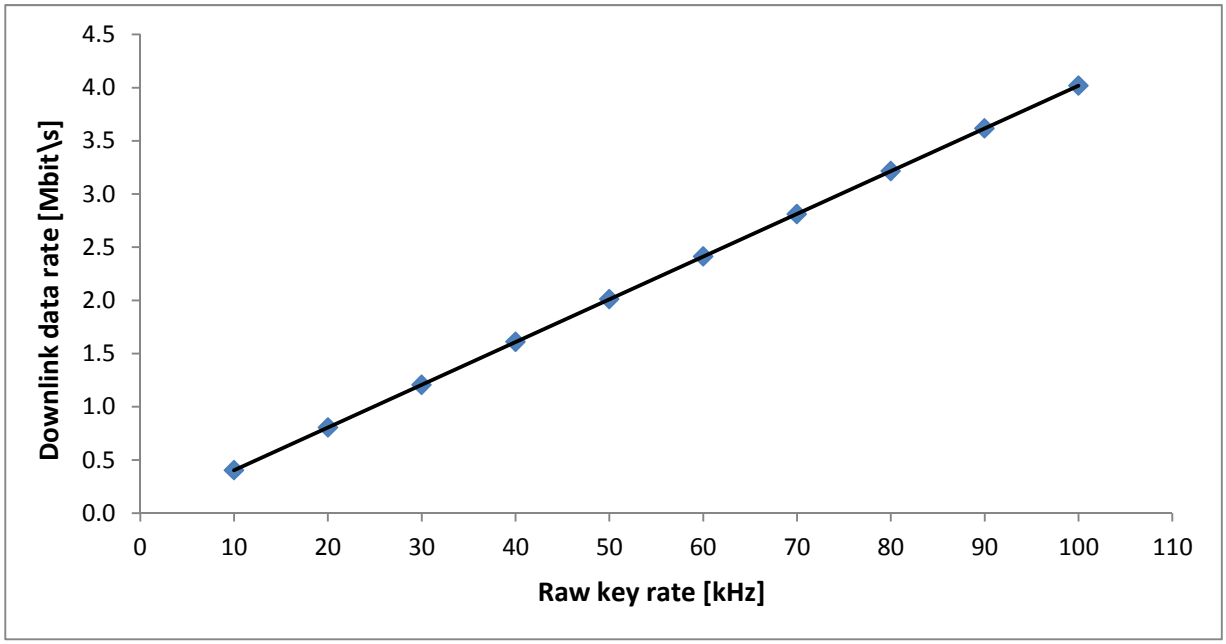

Figure 4.1: Downlink classical communication data rates for different raw key rates.

Raw key rate (Hz)	Coincidence data (byte)	EC data (byte)	PA data (byte)	Total (byte)	Data rate (kbit/s)
10,000	369,915	20,400	24,000	414,315	11
20,000	738,671	20,400	49,500	808,571	22
30,000	1,107,418	20,400	72,300	1,200,118	32
40,000	1,476,168	20,400	98,100	1,594,668	43
50,000	1,844,910	20,400	122,400	1,987,710	53
60,000	2,213,661	20,400	144,600	2,378,661	63
70,000	2,582,413	20,400	171,600	2,774,413	74
80,000	2,951,164	20,400	192,000	3,163,564	84
90,000	3,319,908	20,400	216,900	3,557,208	95
100,000	3,688,662	20,400	243,300	3,952,362	105

Table 4.9: Uplink (ground to satellite) communication statistics resulting from processing 300 seconds of QKD data.

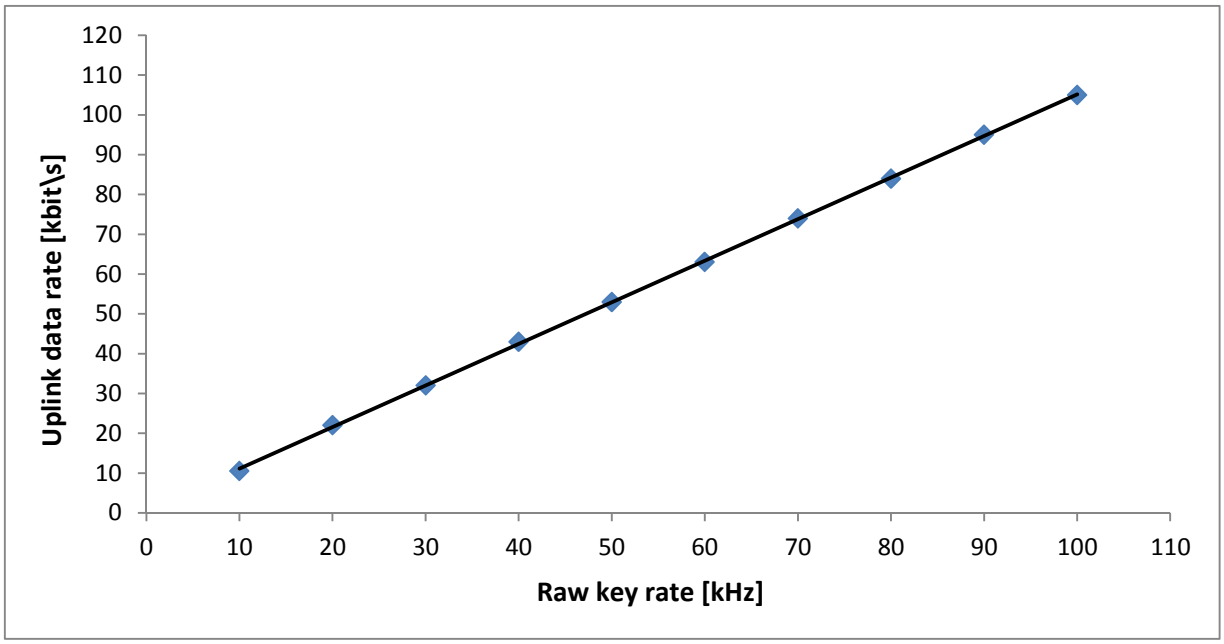

Figure 4.2: Uplink classical communication data rates for different raw key rates.

Chapter 5

Experimental Results

This chapter discusses the performance of each component of our high-loss QKD system as well as our QKD results. Sections 5.1, 5.2 and 5.3 examine the performance of the WCP source, the quantum receiver and the satellite-side software component respectively. Section 5.4 presents our QKD rates obtained at various loss regimes.

5.1 WCP Source Performance

The performance of our WCP source was first characterized in [16, 54]. However, the experimental apparatus has since evolved as discussed in chapter 2. In the previous version of the experiment [16, 54], polarization measurements were performed one measurement basis at a time due the lack of full quantum receiver, which has been added recently (section 2.3). Moreover, here, the FPGA-controlled modulator (section 2.2.1) is modified so that it produces a 128-state sequence of signal and decoy states with a different profile than the 256-state sequence used in [16, 54]. To verify the quality of the source output, we take measurements to ensure that the new sequence exhibits the desired signal/decoy levels and state visibilities.

To measure the signal/decoy levels, we analyze the timing information of photon detections coming from Alice's fiber splitter (see figure 2.3). The resulting histogram is displayed in figure 5.1. The histogram is produced by binning the time differences between time-tags of detection events and the time-tag of the nearest reference pulse (which marks the beginning of a new sequence) coming from the modulator. Signal and decoy states are clearly identifiable as shorter (under 2×10^5 counts) or longer bars in the histogram, which correspond to the two different values of average photon number per pulse ($\mu = 0.5$

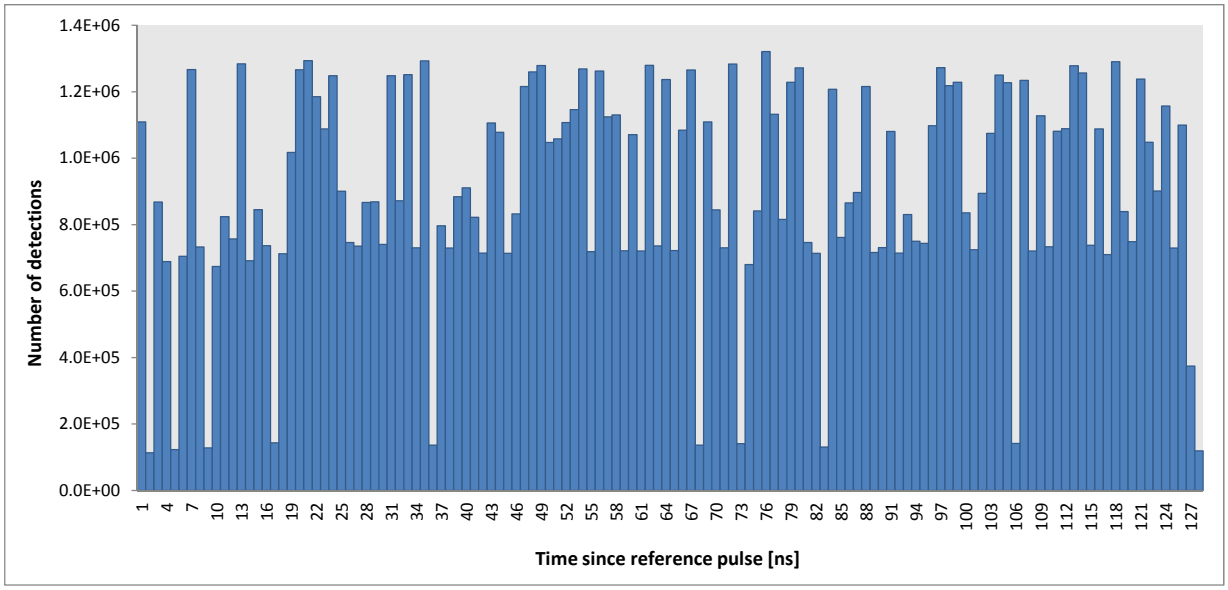

Figure 5.1: Timing histogram of photon detections coming from the fiber splitter at the WCP source. Decoy states are clearly identifiable as shorter bars (under 2×10^5 counts) in the histogram, while the taller bars correspond to signal states in the sequence. The counts in time bin 127 correspond to a special *reset* state produced by the modulator to delimit consecutive sequences.

and $\nu = 0.1$). The measured locations of decoy states in the histogram agree with the modulation sequence programmed into the modulator.

We use the quantum receiver (section 2.3) to measure the visibilities of the polarization states produced by the source. When the 810 nm Ti:Sapph laser is in continuous wave mode (i.e. not model-locked), the output photons at 532 nm have visibility of over 99% in both bases. In the normal mode-locked state, the modulator is driven at the Ti:Sapph laser frequency of 76 MHz. In this regime, visibility of >98% is maintained in the rectilinear (H/V) measurement basis and >95% in the diagonal (D/A) basis.

Figure 5.2 displays the temporal variation of the QBER[1] averaged over both bases. We can see that the QBER is stable over long periods of several hours (it averages at $1.8 \pm 0.9\%$ in the last 160 minutes) and it only degrades due to changing room temperature/humidity in the lab. A large temperature gradient affects the birefringence in the many optical fibers which leads to the misalignment of the polarization reference frames of the source and the receiver (see section 3.5).

[1]QBER and visibility are linearly related: $\text{QBER} = \frac{1-\text{Visibility}}{2}$

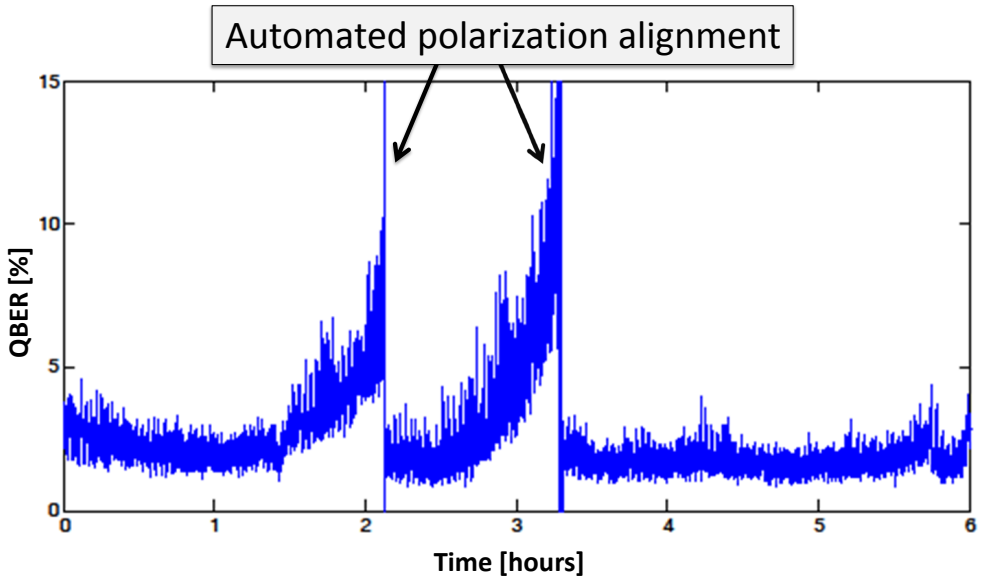

Figure 5.2: Stability measurement of the overall QKD system-wide QBER over a prolonged period of time (6 hours) at 25 dB total channel loss. When the QBER rises significantly due to changing lab temperature/humidity and laser spectrum drifts, we apply the automated polarization procedure (section 3.5) to correct the alignment and reduce the QBER.

5.2 Quantum Receiver Performance

The performance of the quantum receiver is initially tested with polarized light produced by placing a polarizer in front of the output beam of a strong laser at 532 nm and shining this beam at the receiver through a neutral density (ND) filter to reduce the intensity. Our tests show that the visibility is high ($> 98\%$) in all four polarization states considered for the BB84 protocol (H, V, D and A).

Due to the experimental nature of the optomechanical components utilized, some occasional fine adjustments of the alignment of the optical path are required to maintain high visibility. A professionally machined rigid framework would not suffer from this problem. Nevertheless, the receiver's alignment has been observed to be stable over several days.

Furthermore, the quantum receiver is tested as part of the overall high-loss QKD system using the WCP source described in section 2.2, the QKD software detailed in chapter 3 and the polarization alignment software[2]. As illustrated in figure 5.2, our automated polarization alignment procedure works very well to improve the QBER in the case of alignment drifts between the polarization reference frames of the source and the receiver.

[2]As discussed in section 3.5, the quantum receiver contains the wave plate triplet which is used for our automated polarization alignment procedure.

Raw key rate (Hz)	Sifted key rate (Hz)	QBER (%)	Processing time (sec)	OS overhead (sec)	Memory usage (Mbyte)
10,000	3,538	4.4	46.7	14.4	25.98
20,000	7,186	4.8	65.4	16.2	43.06
30,000	10,586	4.6	86.7	18.3	59.11
40,000	13,833	4.9	115.9	18.4	75.63
50,000	17,512	5.0	157.1	21.5	93.74
60,000	21,145	4.9	206.1	21.8	110.30
70,000	24,552	4.8	257.7	23.5	125.38
80,000	28,276	4.7	323.5	24.6	141.92
90,000	32,489	4.8	408.3	26.6	158.44
100,000	35,527	5.1	481.9	29.2	175.04

Table 5.1: Computation statistics resulting from processing 300 seconds of QKD data. The processing time, OS overhead and memory usage of the satellite-side QKD process have been measured with the Linux *time* command. The operating system (OS) overhead is the time taken up by OS-level facilities invoked by the QKD process.

5.3 Satellite-Side Software Performance

The satellite-side software component is tested on an inexpensive ($150), low-power (2 W) embedded system, namely the Freescale IMX53 QSB single-board computer. This board features a single-core, 1 GHz ARM processor with 1 Gbyte of RAM and standard 100 Mbit Ethernet connectivity [120]. The measured performance, displayed in figure 5.3, is in line with our expectations and the computing resource requirements detailed in chapter 4. In fact, the current bottleneck is not at Bob's side but rather at the computer on Alice's side and at the network communication link between Alice and Bob. In a future implementation, the computer/server at Alice can be made significantly more powerful than our current desktop PC.

As discussed in section 4.2, we use the weak coherent source and quantum receiver to collect experimental QKD data for 300 seconds at a receiver detection rate of about 150 kHz. Each one second chunk of this data is then truncated to produce lower key rates in the range we expect for satellite-based QKD [18]. The full QKD protocol is performed on the resulting data subsets at predefined raw-key rates to obtain computation statistics for a range of rates. The focus in this section is on software performance and not on QKD rates, which are the topic of section 5.4.

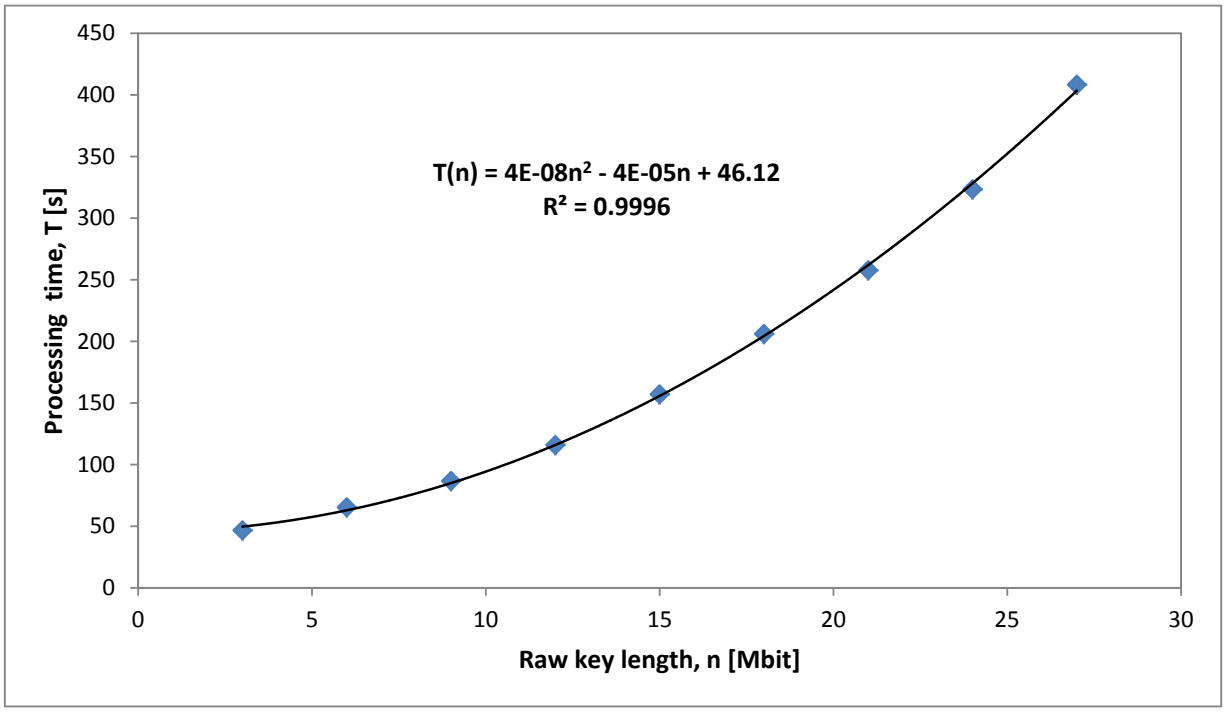

Figure 5.3: Performance of the satellite-side QKD process running on a Freescale IMX53 embedded ARM board. The processing time scales quadratically with the raw-key rate. The R^2 value of 0.9996 indicates a very strong correlation, that is, a very good fit of the trend line. This data used for software performance evaluation only. In reality, we do not expect the raw key length to exceed 10 megabits over a single satellite pass.

Table 5.1 shows detailed memory and CPU usage for the satellite-side QKD process. The runtime is graphically depicted in figure 5.3. As seen in section 4.1.2, privacy amplification is asymptotically quadratic in the block size/raw-key length. All the other post-processing steps behave linearly. Hence, it is expected that the performance of the overall QKD process scales quadratically with the raw-key length as observed in figure 5.3. Note that figure 5.3 explores very large raw key length values (over 25 Mbit) for scalability purposes. In reality, we do not expect the raw key length to exceed 10 megabits over a single satellite pass [18].

5.4 QKD Results

We test the performance of the entire high-loss QKD system (detailed in chapter 2) comprising the WCP source (section 2.2), the quantum receiver (section 2.3) and the QKD post-processing system (chapter 3) including the polarization alignment software (section 3.5). We measure the QKD rates for various fixed losses above 28 dB, and with channel loss continuously varied to emulate a satellite pass.

Note that the previous version of the experiment [16] provides similar results, but [16] only infers the performance of the QKD protocol in these high-loss conditions by employing a different coincidence analysis and without implementing the error-correction and privacy amplification steps of the protocol. In this experiment, our aim is to use the newly constructed quantum receiver to demonstrate full QKD in action. Similarly to [16], our primary focus is on high-loss regimes which are expected for QKD with a satellite uplink as shown in link analysis and simulations [18].

5.4.1 QKD at Fixed Loss Levels

Table 5.2 summarizes our QKD results for fixed total channel loss between 28.9 dB and 56.1 dB. The secure key rates, plotted in figure 5.4, vary between 2 kbit/s at 28.9 dB and 1 bit/s at 56.1 dB. The coincidence window for each measurement is chosen according to the optimal curve given in figure 5 in [16].

The QBER values tend to increase with channel loss due to lower signal-to-noise (SNR) ratio. However, the QBER decreases as the size of the coincidence window gets smaller, because of better temporal filtering—smaller coincidence window means less background photon detection events are accidentally included in the sifted key.

Secure key lengths are calculated through the weak+vacuum decoy-state asymptotic key rate formalism [14] discussed in section 1.2.2, including the error correction efficiency, $\eta_{\mathrm{EC}} \in [1.3, 1.7]$, achieved by the LDPC algorithm (see section 3.6.2). Note that [16] assumes the use of the Cascade error correction protocol (see section 3.6.1) with a fixed $\eta_{\mathrm{EC}} = 1.22$, and hence, the secure key rates reported there are slightly higher.

5.4.2 Emulating a Satellite Pass

To emulate a satellite pass, the loss in the quantum channel is slowly and continuously varied with the movable lens (figure 2.4) from 60 dB to 25 dB and back. Then, a curve fit of the per-second loss data is produced and points from that curve are systematically

Loss (dB)	QBER (%)	Coincidence window (ns)	Sifted key rate (bit/s)	Secure key rate (bit/s)	Secure key rate (bit/laser pulse)
28.9	2.9	1.60	16,659	2,015	6.24×10^{-5}
34.8	2.6	1.50	3,064	538	1.78×10^{-5}
40.1	2.2	1.40	1,145	159	6.58×10^{-6}
45.4	2.7	1.25	393	56	1.70×10^{-6}
50.1	4.3	1.20	119	17	5.17×10^{-7}
52.0	3.9	0.70	49	8	2.60×10^{-7}
54.0	2.8	0.40	18	7	1.92×10^{-7}
56.1	2.1	0.30	8	1	2.71×10^{-8}

Table 5.2: Sifted and secure key rates and associated statistics for a range of fixed loss levels. The raw key rate (not shown) is about double the sifted key rate. The coincidence window is chosen according to the optimal curve given in figure 5 in [16]. Note that the QBER increases with channel loss and decreases with coincidence window size.

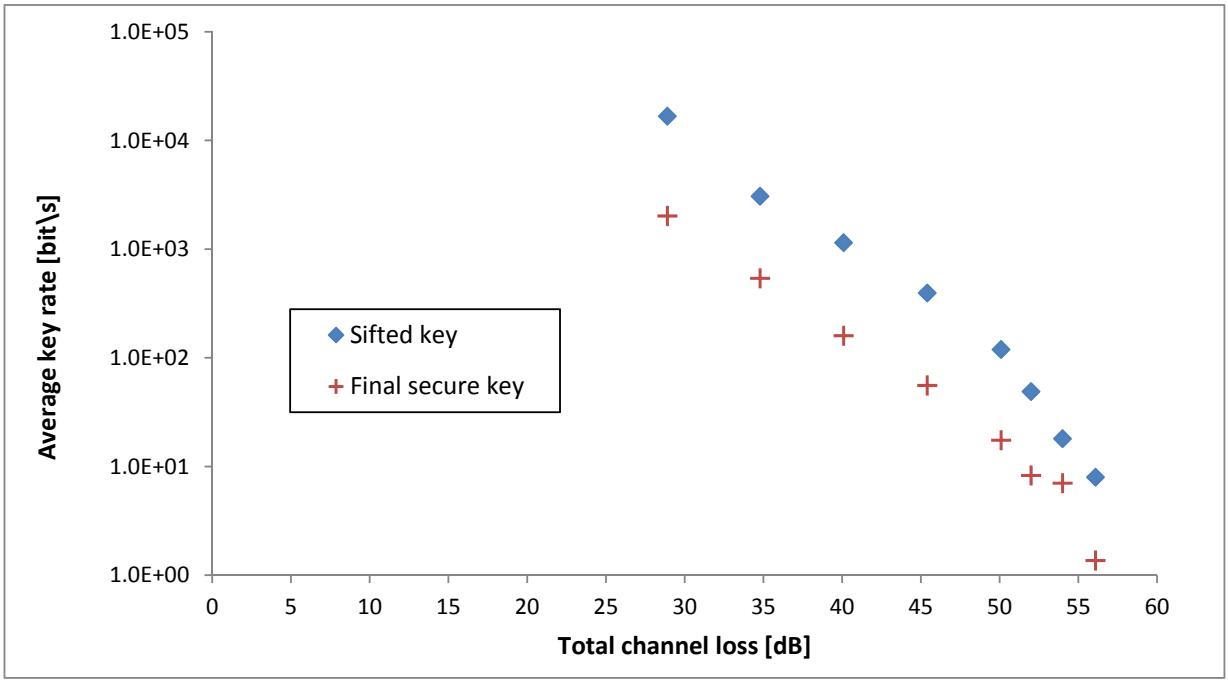

Figure 5.4: Average sifted and secure key rates versus total channel loss in high-loss regimes.

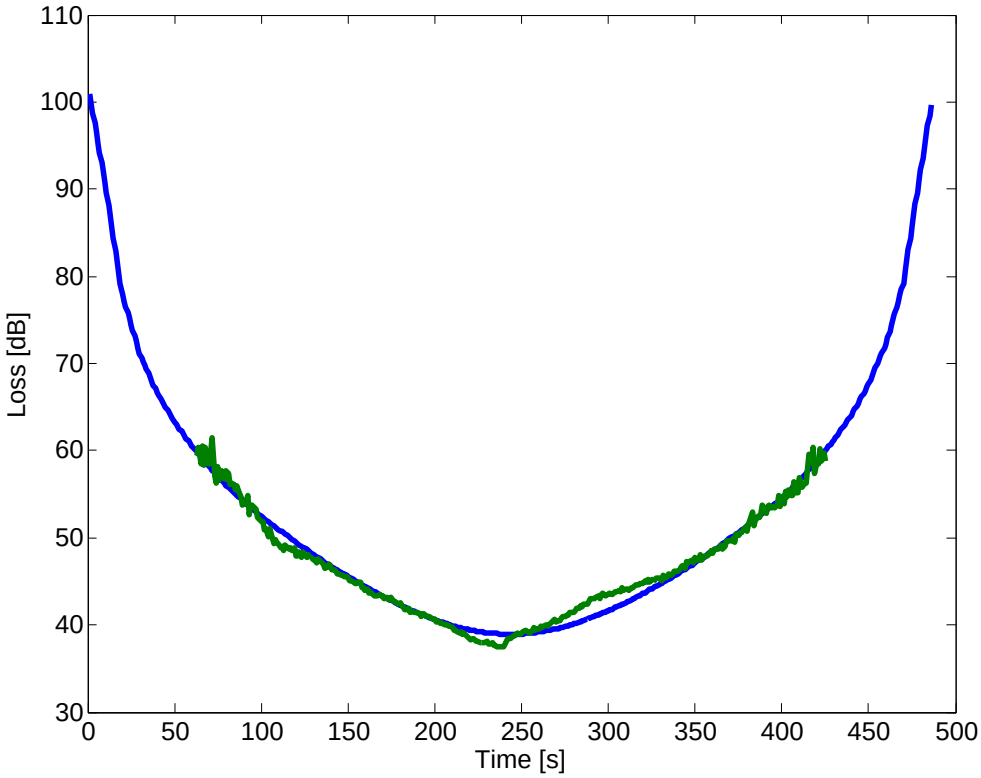

Figure 5.5: Loss curves for an upper quartile [18] satellite pass (blue) and our experimental data (green).

selected to closely match the expected loss curve (blue in figure 5.5) of an upper-quartile satellite pass [18]. The resulting loss curve is displayed in green in figure 5.5.

Each selected point corresponds to a one-second chunk of QKD data collected during the long measurement with variable loss. Those one-second chunks are then put together to produce the green loss curve in figure 5.5. Effectively, we piece together multiple chunks of QKD data (taken at various loss levels) to produce a continuous data set which closely resembles QKD data from a satellite uplink.

With our 76 MHz WCP source, assuming poor atmospheric conditions, and including finite-size effects without statistical fluctuations [14] (see section 1.2.2), we obtain 16.08 kbit of secure key for a typical good (upper quartile) pass [18, 49].

Chapter 6

Conclusion and Outlook

The latest advances in computing and communications technology have come with a growing demand for security and privacy. Quantum key distribution (QKD) promises to deliver unconditional security guaranteed by the fundamental laws of physics; however, its point-to-point range is currently limited to terrestrial links of up to 260 km.

The research presented in this thesis contributes towards a potential global-scale deployment of QKD over free-space links to an orbiting satellite. As part of ongoing feasibility studies of the Canadian Quantum Encryption and Science Satellite (QEYSSat) mission, this work focuses on the necessary data processing algorithms and their computing resource requirements on the spacecraft. It also details the design, implementation and performance analysis of a complete QKD testbed system employing a decoy-state BB84 scheme under high loss to demonstrate the feasibility of QKD with a satellite uplink. Our experiment helps to establish good estimates of the overall complexity, the computing resources necessary, and the bandwidth requirements of the RF links. Thus, it provides a foundation for the future development of the quantum payload onboard QEYSSat.

The QKD post-processing subsystem described in this thesis aims to minimize the computing requirements at one side of the link, unlike most traditional implementations which assume symmetric computing resources at each end. It features precise coincidence analysis, error correction based on low-density parity-check codes, privacy amplification employing Toeplitz hash functions, and a procedure for automated polarization alignment.

Our main result is that the necessary processing power and memory requirements are well within the capabilities of modern low-power computing technology. We estimate that a CPU with at least 750 MHz clock is needed on the satellite and at least 256 Mbyte of memory. We conclude that, in the worst case, an RF link with 5 Mbit of downlink and 100 kbit of uplink bandwidth is required. As a proof of concept, we ran the satellite-side

QKD software on an ARM board (consuming 2 W of power) and measured its performance while executing a full QKD protocol. Our performance results are promising and our implementation is platform-independent, so it can be ported to any space-qualified computing system.

Furthermore, the entire QKD system was tested at a range of fixed high losses and was able to generate a secure key at up to 56.1 dB of loss in the quantum channel. We also performed tests under varying-loss conditions similar to those in a real satellite uplink. We obtained 16.08 kbit of secure key for an upper-quartile pass, with a communication time of less than 5 minutes, a realistic duration of a satellite key exchange.

In the future, a number of areas in our current satellite-based QKD testbed system could be improved:

- *WCP Source*: Our Ti:Sapph laser has a repetition rate of 76 MHz, however, Ti:Sapph lasers operating at several GHz have recently become available [121]. Since the modulation subsystem is capable of running at those frequencies, the WCP source can in principle be upgraded with a simple replacement of the current Ti:Sapph laser.

- *Modulator*: The current pseudo-random sequence should be replaced with a truly random one which never repeats.

- *Receiver*: The compensation components of the automated polarization alignment system should be moved from the receiver to the source. The overall stability of the receiver should be improved, and the receiver should be tested outdoors preferably on a moving platform.

- *Data Processing Software*: Hardware-specific acceleration techniques such as SIMD extensions could improve the performance results by a factor of three to four. The coincidence algorithm should be adapted to work with a moving receiver. GPU acceleration could be used for LDPC decoding on the ground. The optimized version of the privacy-amplification algorithm with FFTs should be implemented.

- *Data Processing Hardware*: A fully integrated prototype of the payload data processing system should be developed. The computer could be based on an existing space-qualified SOC such as the REACT card by Neptec Technologies [122] featuring a 1.2 GHz PowerPC MPC8548, 512 MB of DDR2 RAM with built-in error correction, as well as an on-board FPGA that can be used to port DotFast's existing time-tagging solution.

These improvements will further demonstrate the technological readiness of satellite-based QKD and bring the QEYSSat proposal closer to a concrete mission in space.

References

[1] Scarani, V. *et al.* The security of practical quantum key distribution. *Rev. Mod. Phys.* **81**, 1301–1350 (2009).

[2] Shannon, C. E. Communication theory of secrecy systems. *Bell System Technical Journal* **28**, 657 (1949).

[3] Wootters, W. & Zurek, W. A single quantum cannot be cloned. *Nature* **299**, 802 (1982).

[4] Stebila, D., Mosca, M. & Lütkenhaus, N. The case for quantum key distribution. *arXiv:0902.2839 [quant-ph]* (2009).

[5] Shor, P. In *35th Annual Symposium on the Foundations of Computer Science* (1994).

[6] Rivest, R. L., Shamir, A. & Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**, 120–126 (1978).

[7] Stucki, D. *et al.* High rate long distance quantum key distribution over 250 km of ultra low loss fibres. *New Journal of Physics* **11**, 075003 (2009).

[8] Schmitt-Manderbach, T. *et al.* Experimental demonstration of free-space decoy-state quantum key distribution over 144 km. *Phys. Rev. Lett.* **98**, 010504 (2007).

[9] Grosshans, F. & Grangier, P. Continuous variable quantum cryptography using coherent states. *Phys. Rev. Lett.* **88**, 057902 (2002).

[10] Fossier, S., Diamanti, E., Debuisschert, T., Tualle-Brouri, R. & Grangier, P. Field test of a continuous-variable quantum key distribution prototype. *New. J. Phys* **11**, 045023 (2009).

[11] Brassard, G., Lütkenhaus, N., Mor, T. & Sanders, B. C. Limitations on practical quantum cryptography. *Phys. Rev. Lett.* **85**, 1330–1333 (2000).

[12] Hwang, W.-Y. Quantum key distribution with high loss: Toward global secure communication. *Phys. Rev. Lett.* **91**, 057901 (2003).

[13] Lo, H.-K., Ma, X. & Chen, K. Decoy state quantum key distribution. *Phys. Rev. Lett.* **94**, 230504 (2005).

[14] Ma, X., Qi, B., Zhao, Y. & Lo, H.-K. Practical decoy state for quantum key distribution. *Phys. Rev. A* **72**, 012326 (2005).

[15] Yan, Z. *et al.* Novel high-speed polarization source for decoy-state BB84 quantum key distribution over free space and satellite links. *arXiv:1211.3194 [quant-ph]* (2012).

[16] Meyer-Scott, E. *et al.* How to implement decoy-state quantum key distribution for a satellite uplink with 50-dB channel loss. *Phys. Rev. A* **84**, 062326 (2011).

[17] Cai, R. Y. Q. & Scarani, V. Finite-key analysis for practical implementations of quantum key distribution. *New Journal of Physics* **11**, 045024 (2009).

[18] Bourgoin, J. *et al.* A study on satellite quantum key distribution. *In Preparation* .

[19] Scarani, V. & Renner, R. Quantum cryptography with finite resources: Unconditional security bound for discrete-variable protocols with one-way postprocessing. *Phys. Rev. Lett.* **100**, 200501 (2008).

[20] Tan, Y. G. & Cai, Q. Y. Practical decoy state quantum key distribution with finite resource. *The European Physical Journal D* **56**, 449–455 (2010).

[21] Sun, S.-H., Liang, L.-M. & Li, C.-Z. Decoy state quantum key distribution with finite resources. *Physics Letters A* **373**, 2533–2536 (2009).

[22] Li, H.-W. *et al.* Security of decoy states qkd with finite resources against collective attacks. *Optics Communications* **282**, 4162 – 4166 (2009).

[23] Zhao, Y., Qi, B., Ma, X., Lo, H.-K. & Qian, L. Simulation and implementation of decoy state quantum key distribution over 60km telecom fiber. *Proc. of 2006 IEEE ISIT* 2094–2098 (2006).

[24] ID Quantique. URL http://www.idquantique.com.

[25] MagiQ Technologies. URL http://www.magiqtech.com.

[26] Takesue, H. *et al.* Quantum key distribution over a 40-db channel loss using superconducting single-photon detectors. *Nat Photon* **1**, 343–348 (2007).

[27] Yan, Z. *et al.* An ultra low noise telecom wavelength free running single photon detector using negative feedback avalanche diode. *Rev. Sci. Inst.* **83**, 073105 (2012).

[28] Ursin, R. *et al.* Free-space distribution of entanglement and single photons over 144 km. *Nature Physics* **3**, 481–486 (2007).

[29] Briegel, H.-J., Dür, W., Cirac, J. I. & Zoller, P. Quantum repeaters: The role of imperfect local operations in quantum communication. *Phys. Rev. Lett.* **81**, 5932 (1998).

[30] Sangouard, N., Simon, C., de Riedmatten, H. & Gisin, N. Quantum repeaters based on atomic ensembles and linear optics. *Rev. Mod. Phys.* **83**, 33–80 (2011).

[31] Sasaki, M. *et al.* Field test of quantum key distribution in the tokyo qkd network. *Opt. Express* **19**, 10387–10409 (2011).

[32] Peev, M. *et al.* The SECOQC quantum key distribution network in Vienna. *NJP* **11**, 075001 (2009).

[33] Buttler, W. T. *et al.* Daylight quantum key distribution over 1.6 km. *PRL* **84**, 5652 (2000).

[34] Gilbert, G. & Hamrick, M. Practical quantum cryptography: A comprehensive analysis (part one). *arXiv:quant-ph/0009027* (2000).

[35] Nordholt, J. E., Hughes, R. J., Morgan, G. L., Peterson, C. G. & Wipf, C. C. Present and future free-space quantum key distribution. In Mecherle, G. S. (ed.) *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 4635 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, 116–126 (2002).

[36] Rarity, J. G., Tapster, P. R., Gorman, P. M. & Knight, P. Ground to satellite secure key exchange using quantum cryptography. *New Journal of Physics* **4**, 82 (2002).

[37] Miao, E.-L., Han, Z.-F., Zhang, T. & Guo, G.-C. The feasibility of geostationary satellite-to-ground quantum key distribution. *Physics Letters A* **361**, 29–32 (2007).

[38] Tomaello, A., Bonato, C., Deppo, V. D., Naletto, G. & Villoresi, P. Link budget and background noise for satellite quantum key distribution. *Advances in Space Research* **47**, 802–810 (2011).

[39] Etengu, R. *et al.* Performance comparison of BB84 and B92 satellite-based free space quantum optical communication systems in the presence of channel effects. *J. Opt. Commun.* **32**, 37 (2011).

[40] Nauerth, S. *et al.* Air to ground quantum key distribution. In *QCRYPT2012 Conference, Singapore* (2012).

[41] Wang, J.-Y. *et al.* Direct and full-scale experimental verifications towards ground-satellite quantum key distribution. *arXiv:1210.7556* (2012).

[42] Aspelmeyer, M., Jennewein, T., Pfennigbauer, M., Leeb, W. R. & Zeilinger, A. Long-distance quantum communication with entangled photons using satellites. *IEEE Journal of Selected Topics in Quantum Electronics* **9**, 1541–1551 (2003).

[43] Bennett, C. H., Brassard, G. & Mermin, N. D. Quantum cryptography without bell's theorem. *Phys. Rev. Lett.* **68**, 557–559 (1992).

[44] Ling, A. *et al.* Experimental quantum key distribution based on a bell test. *Phys. Rev. A* **78**, 020301 (2008).

[45] Lo, H.-K., Curty, M. & Qi, B. Measurement-device-independent quantum key distribution. *Phys. Rev. Lett.* **108**, 130503 (2012).

[46] Ursin, R. *et al.* Space-QUEST: Experiments with quantum entanglement in space. In *IAC Proceedings A2.1.3* (2008).

[47] Xin, H. Chinese academy takes space under its wing **332**, 904 (2011).

[48] Takenaka, H., Toyoshima, M., Takayama, Y., Koyama, Y. & Akioka, M. Experiment plan for a small optical transponder onboard a 50 kg-class small satellite. 113 (2011).

[49] Higgins, B. L. *et al.* Detailed performance analysis of the proposed QEYSSAT quantum receiver satellite. In *Quantum Electronics and Laser Science Conference (QELS)* (2012).

[50] D'Souza, I. *et al.* The QEYSSat mission: Demonstrating global quantum key distribution using a microsatellite. In *SSC12-IX-1, Small Satellite Conference* (2012).

[51] Accetta, J. S. & Shumaker, D. L. *The Infrared and electro-optical systems handbook*, vol. 2 (Infrared Information Analysis Center and SPIE Optical Engineering Press, 1993).

[52] Rideout, D. *et al.* Fundamental quantum optics experiments conceivable with satellites—reaching relativistic distances and velocities. *Classical and Quantum Gravity* **29**, 224011 (2012).

[53] Sun, X., Krainak, M. A., Abshire, J. B. & Spinhirne, J. D. Space-qualified silicon avalanche-photodiode single-photon-counting modules. *J. Mod. Opt.* **51**, 1333–1350 (2004).

[54] Meyer-Scott, E. *Experimental quantum communication in demanding regimes*. Master's thesis, University of Waterloo (2011).

[55] Dixon, A. R., Yuan, Z. L., Dynes, J. F., Sharpe, A. W. & Shields, A. J. Gigahertz decoy quantum key distribution with 1 mbit/s secure key rate. *Opt.Express* **16**, 18790–18797 (2008).

[56] Dixon, A. R., Yuan, Z. L., Dynes, J. F., Sharpe, A. W. & Shields, A. J. Continuous operation of high bit rate quantum key distribution. *Applied Physics Letters* **96**, 161102 (2010).

[57] Jofre, M. *et al.* 100 mhz amplitude and polarization modulated optical source for free-space quantum key distribution at 850 nm. *J. Lightwave Technol.* **28**, 2572–2578 (2010).

[58] Jofre, M. *et al.* Fast optical source for quantum key distribution based on semiconductoroptical amplifiers. *Opt. Express* **19**, 3825–3834 (2011).

[59] Jennewein, T., Simon, C., Weihs, G., Weinfurter, H. & Zeilinger, A. Quantum cryptography with entangled photons. *Phys. Rev. Lett.* **84**, 4729–4732 (2000).

[60] Tanzilli, S. *et al.* Ppln waveguide for quantum communication. *The European Physical Journal D - Atomic, Molecular, Optical and Plasma Physics* **18**, 155–160 (2002).

[61] Meyer-Scott, E. *et al.* Quantum entanglement distribution with 810 nm photons through telecom fibers. *Applied Physics Letters* **97**, 031117 (2010).

[62] Ma, X., Fung, C.-H. F. & Lo, H.-K. Quantum key distribution with entangled photon sources. *Physical Review A* **76**, 012307 (2007).

[63] Erven, C., Ma, X., Laflamme, R. & Weihs, G. Entangled quantum key distribution with a biased basis choice. *New Journal of Physics* **11**, 045025 (15pp) (2009).

[64] Gottesman, D., Lo, H.-K., Lütkenhaus, N. & Preskill, J. Security of quantum key distribution with imperfect devices. *Quantum Information and Computation* **4**, 325–360 (2004).

[65] Hadfield, R. H. Single-photon detectors for optical quantum information applications. *Nat Photon* **3**, 696–705 (2009).

[66] Boyd, R. W. *Nonlinear Optics*, chap. 3 (London : Academic Press, 2003), 2nd edn.

[67] URL http://www.thorlabs.de/navigation.cfm?guide_id=2002.

[68] URL http://www.newport.com/XPS-Series-Universal-High-Performance-Motion-Cont/300904/1033/info.aspx.

[69] URL http://www.micro-photon-devices.com/.

[70] MacKay, D. J. C. & Neal, R. M. Near shannon limit performance of low density parity check codes. *Electronics Lett.* **33**, 457–458 (1997).

[71] Pearson, D. High-speed qkd reconciliation using forward error correction. In *Proc. 7th International Conference on Quantum Communication, Measurement and Computing (QCMC)*, 299–302 (2004).

[72] Martinez, I. L., Chan, P., Mo, X., Hosier, S. & Tittel, W. Proof-of-concept of real-world quantum key distribution with quantum frames. *New J. Phys.* **11**, 095001 (2009).

[73] Krawczyk, H. LFSR-based hashing and authentication. In Desmedt, Y. (ed.) *Advances in Cryptology — CRYPTO '94*, vol. 839 of *Lecture Notes in Computer Science*, 129–139 (Springer Berlin / Heidelberg, 1994).

[74] MSDN. URL http://msdn.microsoft.com/en-us/library/system.threading.tasks.task.aspx.

[75] Semenenov, A. A. & Vogel, W. Quantum light in the turbulent atmosphere. *Phys. Rev. A* **80**, 021802 (2009).

[76] Toyoshima, M. *et al.* Polarization measurements through space-to-ground atmospheric propagation paths by using a highly polarized laser source in space. *Opt. Express* **17**, 22333–22340 (2009).

[77] Kumar, A. & Ghatak, A. *Polarization of Light with Applications in Optical Fibers* (SPIE Press, 2011).

[78] Treiber, A. *et al.* A fully automated entanglement-based quantum cryptography system for telecom fiber networks. *New Journal of Physics* **11**, 045013 (2009).

[79] Altepeter, J. B., James, D. F. V. & Kwiat, P. G. Qubit quantum state tomography. *Lect. Notes Phys.* **649**, 113–145 (2004).

[80] Gottesman, D. *Stabilizer codes and quantum error correction.* Ph.D. thesis, California Institute of Technology (1997).

[81] Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information (Cambridge Series on Information and the Natural Sciences)* (Cambridge University Press, 2010), 10th anniversary edn.

[82] Cover, T. M. & Thomas, J. A. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)* (Wiley-Interscience, 2006).

[83] Shannon, C. E. A mathematical theory of communication. *Bell System Technical Journal* **27**, 379–423 (1948).

[84] Bennett, C., Bessette, F., Brassard, G., Salvail, L. & Smolin, J. Experimental quantum cryptography. *Journal of Cryptology* **5**, 3–28 (1992).

[85] Brassard, G. & Salvail, L. Secret-key reconciliation by public discussion. In *Eurocrypt'93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology, Lecture Notes in Computer Science*, vol. 765, 410–423 (Springer-Verlag, 1994).

[86] Sugimoto, T. & Yamazaki, K. A study on secret key reconciliation protocol Cascade. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Science* **E83-A**, 1987–1991 (2000).

[87] Liu, S., Van Tilborg, H. C. A. & Van Dijk, M. A practical protocol for advantage distillation and information reconciliation. *Des. Codes Cryptography* **30**, 39–62 (2003).

[88] Elkouss, D., Martinez-Mateo, J. & Martin, V. Information reconciliation for quantum key distribution. *Quantum Info. Comput.* **11**, 226–238 (2011).

[89] Buttler, W. T. *et al.* Fast, efficient error reconciliation for quantum cryptography. *Phys. Rev. A* **67**, 052303 (2003).

[90] Han, J. & Qian, X. Auto-adaptive interval selection algorithm for quantum key distribution. *Quantum Info. Comput.* **9**, 693–700 (2009).

[91] Elkouss, D., Leverrier, A., Alléaume, R. & Boutros, J. J. Efficient reconciliation protocol for discrete-variable quantum key distribution. In *Proceedings of the 2009 IEEE international conference on Symposium on Information Theory - Volume 3*, ISIT'09, 1879–1883 (IEEE Press, 2009).

[92] Elkouss, D., Martinez, J., Lancho, D. & Martin, V. Rate compatible protocol for information reconciliation: An application to qkd. In *IEEE Inf. Theory Workshop (ITW)*, 145–149 (2010).

[93] Martinez Mateo, J., Elkouss, D. & Martin, V. Interactive reconciliation with low-density parity-check codes. In *6th Int. Symposium on Turbo Codes & Iterative Information Processing*, 270–274 (2010).

[94] Martinez Mateo, J., Elkouss, D. & Martin, V. Improved construction of irregular progressive edge-growth Tanner graphs. *Communications Letters, IEEE* **14**, 1155 –1157 (2010).

[95] Martinez Mateo, J. *Efficient Information Reconciliation for Quantum Key Distribution*. Ph.D. thesis, Universidad Politecnica de Madrid (2011).

[96] Elliott, C. *et al.* Current status of the DARPA Quantum Network. arXiv:quant-ph/0503058 (2005).

[97] Hu, X.-Y., Eleftheriou, E. & Arnold, D. Regular and irregular progressive edge-growth Tanner graphs. *Information Theory, IEEE Transactions on* **51**, 386 –398 (2005).

[98] Hu, X.-Y., Eletheriou, E. & Arnold, D. Source code for progressive edge growth parity-check matrix construction (2003). URL http://www.inference.phy.cam.ac.uk/mackay/PEG_ECC.html.

[99] Erven, C. *On Experimental Quantum Communication and Cryptography*. Ph.D. thesis, University of Waterloo (2012).

[100] Chan, P. *Low-Density Parity-Check Codes For Quantum Key Distribution*. Master's thesis, University of Calgary (2009).

[101] Gallager, R. G. Low-density parity-check codes. In *Transactions of the IRE Professional Group on Information Theory*, vol. IT-8, 21–28 (1962).

[102] Berrou, C., Glavieux, A. & Thitimajshima, P. Near shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *Communications, 1993. ICC 93. Geneva.*

Technical Program, Conference Record, IEEE International Conference on, vol. 2, 1064 –1070 vol.2 (1993).

[103] SeQureNet. URL http://www.sequrenet.fr/products.html.

[104] Tanner, R. A recursive approach to low complexity codes. In *IEEE Transactions on Information Theory*, vol. 27 (1981).

[105] Luby, M., Amin Shokrolloahi, M., Mizenmacher, M. & Spielman, D. Improved low-density parity-check codes using irregular graphs and belief propagation. In *Information Theory, 1998. Proceedings. 1998 IEEE International Symposium on*, 117 (1998).

[106] Richardson, T. & Urbanke, R. The capacity of low-density parity-check codes under message-passing decoding. *Information Theory, IEEE Transactions on* **47**, 599–618 (2001).

[107] Hu, X.-Y., Eleftheriou, E. & Arnold, D.-M. Irregular progressive edge-growth (PEG) Tanner graphs. In *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, 480 (2002).

[108] Berlekamp, E., McEliece, R. & van Tilborg, H. On the inherent intractability of certain coding problems (corresp.). *Information Theory, IEEE Transactions on* **24**, 384–386 (1978).

[109] Bennett, C. H., Brassard, G. & Robert, J.-M. Privacy amplification by public discussion. *SIAM J. Comput.* **17**, 210–229 (1988).

[110] Bennett, C., Brassard, G., Crepeau, C. & Maurer, U. Generalized privacy amplification. *Information Theory, IEEE Transactions on* **41**, 1915–1923 (1995).

[111] Renner, R. & König, R. *Universally Composable Privacy Amplification Against Quantum Adversaries*, vol. 3378 of *Theory of Cryptography*, 407–425 (Springer, Berlin Heidelberg, 2005).

[112] Shor, P. W. & Preskill, J. Simple proof of security of the BB84 quantum key distribution protocol. *Phys. Rev. Lett.* **85**, 441–444 (2000).

[113] Renner, R. *Security of Quantum Key Distribution*. Ph.D. thesis, Swiss Federal Institute of Technology (2005).

[114] Tsurumaru, T., Matsumoto, W. & Asai, T. QKD post-processing algorithms of mitsubishi electric corporation. AIT QKD Post Processing Workshop 2011 (2011). URL https://sqt.ait.ac.at/software/attachments/download/190.

[115] Gray, R. M. Toeplitz and circulant matrices: A review. *Foundations and Trends in Communications and Information Theory* **2**, 155–239 (2005).

[116] Hayashi, M. Optimal ratio between phase basis and bit basis in quantum key distributions. *Phys. Rev. A* **79**, 020303 (2009).

[117] Hayashi, M. Exponential decreasing rate of leaked information in universal random privacy amplification. *Information Theory, IEEE Transactions on* **57**, 3989–4001 (2011).

[118] Golub, G. H. & Loan, C. F. V. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)* (The Johns Hopkins University Press, 1996).

[119] Fung, C.-H. F., Ma, X. & Chau, H. F. Practical issues in quantum-key-distribution post-processing. *Physical Review A* **81**, 17 (2009).

[120] URL http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=IMX53QSB.

[121] Bartels, A., Heinecke, D. & Diddams, S. A. Passively mode-locked 10 GHz femtosecond Ti:sapphire laser. *Opt. Lett.* **33**, 1905–1907 (2008).

[122] Neptec. URL http://www.neptec.com/technology/space/react.php.

[123] Tomayko, J. E. Computers in spaceflight: The NASA experience. URL http://www.hq.nasa.gov/office/pao/History/computers/contents.html.

[124] Cooper, A. E. & Chow, W. T. Development of on-board space computer systems. *IBM J. Res. Dev.* **20**, 5–19 (1976).

[125] Sterritt, R., Rouff, C. A., Hinchey, M. G., Rash, J. L. & Truszkowski, W. Next generation system and software architectures challenges from future NASA exploration missions. *Science of Computer Programming* **61**, 48–57 (2006).

[126] Bajracharya, M., Maimone, M. & Helmick, D. Autonomy for Mars rovers: Past, present, and future. *Computer* **41**, 44–50 (2008).

[127] Peters, B. *et al.* Flight SEU performance of the single board computer (SBC) utilizing hardware voted commercial PowerPC processors on-board the CALIPSO satellite. In *Radiation Effects Data Workshop, 2007 IEEE*, vol. 0, 16–25 (2007).

[128] Bueno, R. *Performance and Dependability of RapidIO-Based Systems for Real-Time Space Applications.* Ph.D. thesis, University of Florida (2006).

[129] Alkalai, L. & Geer, D. Advanced computing technologies for future deep space exploration missions (2004). URL http://hdl.handle.net/2014/22895.

[130] Alkalai, L. An overview of flight computer technologies for future NASA space exploration missions. *Acta Astronautica* **52**, 857–867 (2003).

[131] Wind River Systems. URL http://www.windriver.com/news/press/pr.html?ID=314.

[132] Paschalidis, N. A family of analog and mixed signal vlsi asics for NASA science missions. In Harris, R. A. (ed.) *Low-Cost Planetary Missions*, vol. 542 of *ESA Special Publication*, 313–319 (2003).

[133] Chau, S., Alkalai, L., Tai, A. & Burt, J. Design of a fault-tolerant COTS-based bus architecture. *IEEE Transactions on Reliability* **48**, 351 –359 (1999).

[134] Space Micro. URL http://www.spacemicro.com/space_div/se_div.htm.

[135] CPU Tech. URL http://www.cputech.com/cpu872.php.

Appendices

Appendix A

A Survey of Computing Hardware Used in Space Applications

Computing hardware has been a key part of spaceflight missions since the early days of space discovery. In fact, the computer industry owes at least part of its early development to the NASA space exploration programs and their challenging data processing requirements [123, 124]. While the first NASA missions relied on low-level, custom logic to perform only the most essential tasks, many modern spacecraft incorporate high-performance, general-purpose computing hardware. Those computers are used for various functions: navigation, attitude control, sensor/scientific instruments data processing, general system control, communication protocols, just to name a few. This transition towards high-performance processing, in-flight software updates and remote-controlled instruments has allowed space missions to become more complex. It has also made them much more flexible, as software changes are a lot cheaper and faster to implement than costly hardware modifications [123].

Another driver for improvement of space computing hardware has been the need for autonomous operations. The risk to human life in manned missions and the remote locations of places of interest (e.g. the round-trip communication delay between Earth and Mars is about 40 minutes) has pushed NASA to focus on unmanned, robotic space explorations. However, autonomous systems such as the recent Mars rovers require a lot of onboard processing power to be able to make complex decisions with stringent real-time constraints, without the possibility of immediate feedback and assistance from mission control on Earth [125, 126].

On the other hand, there are some significant difficulties which hold back the penetration of new technologies into the space exploration domain. Modern computer components

Year	Mission	CPU	RAM	Operating System
1969	Apollo 11	1 MHz Custom, 16-bit	36 KB	Custom
1981	Space Shuttle	4.77 MHz Intel 8086	1 MB	Custom
1997	Sejourner	0.1 MHz Intel 80C85	512 KB	Custom cyclic executive
2004	Spirit/Opportunity	20 MHz IBM RAD6000	128 MB	VxWorks
2006	CALIPSO	160 MHz 603r PowerPC	128 MB	VxWorks
2011	Curiosity	200 MHz IBM RAD750	256 MB	VxWorks

Table A.1: Main computer components in several past NASA missions [123, 124, 126, 127].

as they are cannot be deployed in space because they are not build to withstand all the damaging effects of the harsh environmental conditions (see section A.1). Additional engineering and qualification is required to make those systems more durable and reliable. This process is unfortunately both costly and time-consuming. Manufacturers also have little economical incentive to move forward, as the market for space-grade computers is rather small. Hence, commercial computing solutions on the ground continue to be several technological generations ahead of the ones used in space.

This chapter provides a survey of computing solutions used in space. It is not meant as an extensive study (due to its limited scope), so it mostly refers to NASA-associated projects and missions and omits references to other countries' space programs (published information about those is not very abundant). Section A.2 presents a brief history of computing hardware used by NASA in the past. In section A.3, the transition from custom-built hardware towards partially hardened commercial off-the-shelf components is reviewed as well as the associated fault-tolerant design techniques. Finally, section A.4 gives an overview of what is state-of-the-art in space-grade computer technologies.

A.1 Main Challenges

Space-grade computers face many challenges in the harsh, limited-resource environments which they need to endure. During takeoff, and optionally landing, systems are subject to severe vibrational strain. Once in outer space, temperatures can vary from 4 K to 400 K. Hence, components have to be engineered and built to much higher reliability standards. Usually, this involves using more durable materials and an extensive qualification effort.

Spacecraft also experience various radiation effects. On Earth, the planet's magnetic field and atmosphere provide a natural cosmic radiation shield. In space, high-energy particles can damage sensitive digital electronics (e.g. memory chips), which can render a

spacecraft unusable. The most common are bit-flip errors caused by single-event upsets (SEUs) [128]. Other single event effects such as single-event latchup (SEL), single-event burnout (SEB) and single-event gate rupture (SEGR) are more rare but also more damaging. Unfortunately, modern RAM chips are semiconductor-based, volatile and not very reliable, so for very high radiation environments (e.g. closer to the Sun) older ferrite-core memories are often used. Radiation hardening is the technique used to design and manufacture radiation-resistant hardware components. It consists of shielding sensitive components from electro-magnetic radiation and other forms of interference [128].

Furthermore, space-grade systems need to respect stringent weight, size and power constraints. In fact, a recent study for the Mars Pathfinder mission has shown that reducing the total spacecraft mass, volume and power (by employing newer microelectronics) can not only reduce the overall mission cost but also allow for a more wide-ranging explorations of Mars [129].

By their nature, spacecraft are usually placed in remote locations and not accessible for maintenance and repair, so fault-tolerance, self-diagnostics and remote testability have become critical for a mission's success [123, 125]. The elements of fault-tolerant bus design are examined in section A.3.

A.2 A Brief Historical Overview

For over 50 years, developments in analog and digital electronics have propelled achievements in space explorations. However, for the first 15 years of NASA missions, there are no general purpose computers onboard [123, 130]. In those early stages of space discovery, during the 1960s and 1970s, the main challenge is to design and manufacture basic electronic circuits which could withstand the vibrations of a rocket launch and the hostile conditions in space (as discussed in section A.1). In that time, several research initiatives are created in US government labs to established the requirements and space qualification framework for spacecraft computers. The most notable project is the Self Test And Repair Computer (STAR) project, created at the Jet Propulsion Laboratory (JPL), California Institute of Technology [130]. Interestingly, the concept of fault-tolerance first emerges out of STAR research on reliability of computer systems.

During the 1980s, the trend of exponential growth in commercial computing hardware becomes the norm—validating the so called Moore's law, which predicts that transistor counts on integrated circuits will double every two years. This development pushes NASA to depend more and more on the commercial sector to supply highly reliable spacecraft computers, rather than designing and building custom components and interfaces [123].

One such collaboration with industry, and IBM Federal Systems in particular, gives rise to the Generic Very High Speed Integrated Circuit (GVHSIC) project. The outcome of the initiative is the Common Flight Computer (CFC)—a four-chip, central-control computer deployed on the famous Cassini mission to Saturn and its moon Titan [130].

The strong partnership with IBM leads to a generation of PowerPC-based spacecraft computers in the 1990s and 2000s. IBM Federal Systems becomes Lockheed Martin Federal Systems and several single-board computers are developed for the Mars rover missions: Spirit, Opportunity and Curiosity [126, 130]. As seen in table A.1, those are all based on the IBM RAD architecture—a PowerPC-based architecture designed with radiation hardening in mind. For the first time, the exploration vehicles are using VxWorks—a commercial real-time operating system (RTOS) developed by Wind River Systems [131]. Moreover, all additional software is written in the C programming language. This is a big step forward from the custom cyclic executive operating systems and assembly-code level programming employed on previous missions.

Most recently, NASA/JPL's Center for Integrated Space Microsystems, has been working on a more generic flight control system, the X2000 System Flight Computer (SFC), under the Outer Planets Program, originally intended for the Europa Orbiter Project (meant to launch in 2003 but later delayed to 2008) [130, 132]. X2000 is unique in the sense that its design is purposefully based around cutting-edge commercial off-the-shelf (COTS) components with the goal of improving performance and reducing overall development and manufacturing costs [132, 133]. The X2000 SFC is described in more detail in section A.3.3.

A.3 Transition Towards COTS Components

As mentioned in the previous section, building spacecraft computers with COTS components is a logical, economically justified step. COTS solutions have many advantages: state-of-the-art performance, much lower cost (both immediate and recurring), wide availability of commercial hardware and software components, as well as an established and predictable upgrade path [127, 133].

On the other hand, the challenge with COTS is that such components are not designed and built to high reliability standards. Recently, there have been significant efforts towards developing fault-tolerant computing systems out of unreliable COTS parts mainly by incorporating various forms of redundancy and self-testability into the designs. In the next few sections, we look at two such COTS-based spacecraft computers, the CALIPSO

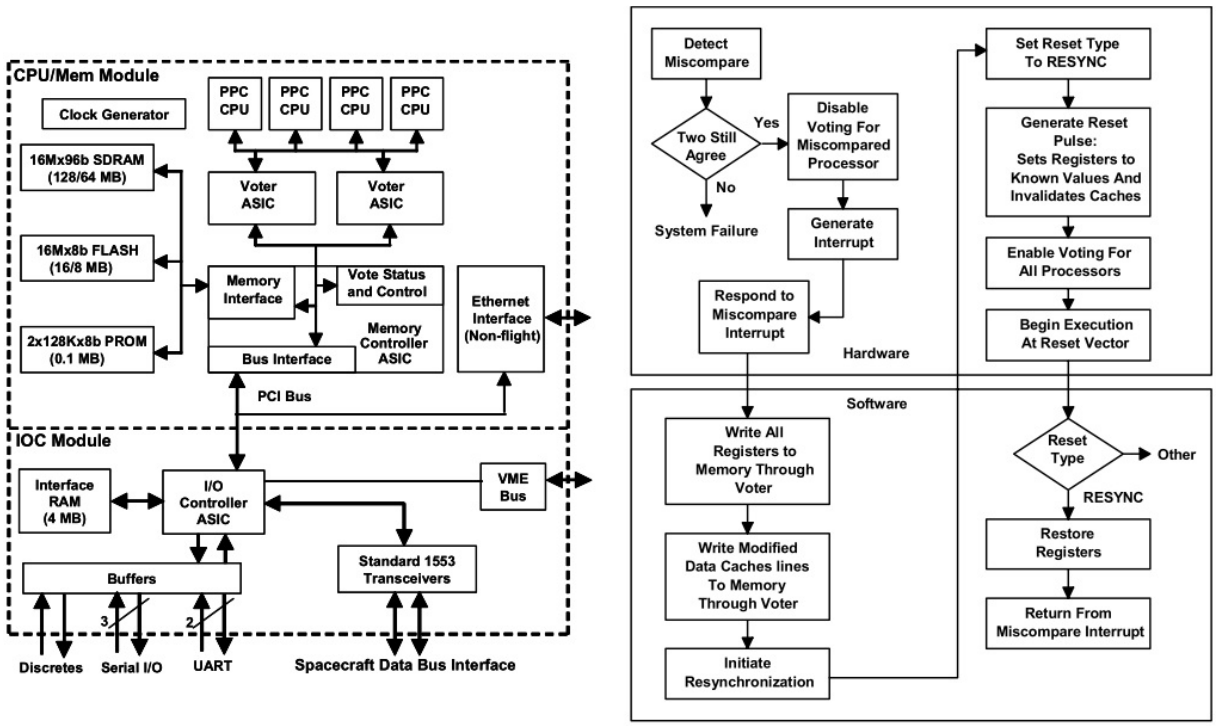

Figure A.1: 4-way voting design (left) and recovery resynchronization logic (right) on the CALIPSO satellite single-board computer [127].

single-board computer (SBC) (section A.3.2) and the JPL X2000 avionics system (section A.3.3).

A.3.1 Common Data Bus Failure Modes

In [133], the authors from NASA/JPL identify the most common critical spacecraft data bus failure modes resulting from radiation effects such as SEUs:

- *Invalid Messages*: Invalid data is found in messages sent over the data bus
- *Non-Responsiveness*: A response to a message returns late (i.e. fails to satisfy its real-time constraint)
- *Babbling*: An uncontrolled data stream causes a communication delay or interruption on the data bus
- *Conflict of Node Address*: Two or more nodes (e.g. processors, micro-controllers, memory blocks, etc.) on the bus have the same address

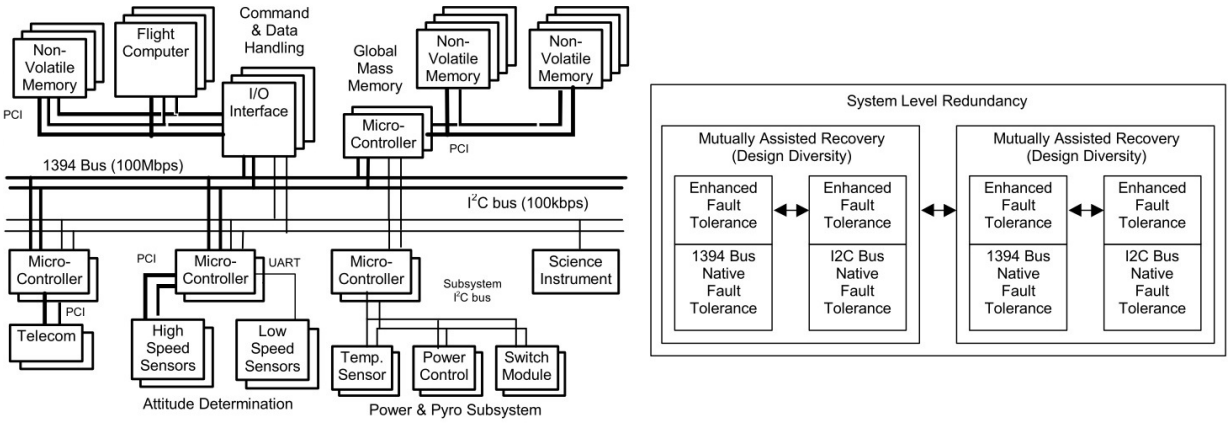

Figure A.2: JPL's X2000 avionics system architecture (left) and multi-layer fault-tolerance strategy (right) [133].

Those common failures are the ones to design against when applying fault-tolerant techniques. According to Chau et al., it is now a standard procedure for NASA/JPL engineers and scientists to perform the so called failure mode effect and criticality analysis (FMECA) for each spacecraft computer design [133].

A.3.2 The CALIPSO Satellite SBC

A good example of a fault-tolerant COTS-based spacecraft computer is the Payload Controller Processor flying onboard the Cloud-Aerosol Lidar and Infrared Pathfinder Satellite Observations (CALIPSO) mission. As per table A.1 the CALIPSO SBC features the 603r PowerPC processor (manufactured by Motorola on the 0.25 μm CMOS/EPI HyperMOS 2.0 process) running at 160 MHz core frequency and the VxWorks RTOS [127]. Redundancy is central to the SBC design employing a 4-way voted processor system as depicted in figure A.1.

The main idea behind the CALIPSO COTS-based SBC design is detecting and amplifying/correcting low level effects at the higher system level. As a result, performance is often traded-off for improved reliability and higher radiation-tolerance [127]. The SBC features the following radiation-effect mitigating design elements [127]:

- *Memory Error Correction and Scrubbing*: Volatile SD-RAM and non-volatile EEPROM are protected with Single-bit Error Correction Double-bit Error Detection

(SECDED). Non-volatile flash memory is redundant (mirrored components) and protected with Reed-Solomon error correcting codes.

- *Hardened Critical Components*: Designated critical system components such as the processor voter ASIC as well as the memory and IO controllers (see figure A.1 left) are radiation hardened.

- *AC/DC Parameter / Frequency Derating*: Several component parameters are derated (or de-tuned) to lower levels. The maximum CPU frequency is reduced to 80% and the SDRAM clock frequency and refresh rates are derated to reduce power, increase reliability and minimize the risk of radiation-induced timing delays [127].

- *Multiple Processor Voting*: As depicted in figure A.1, four COTS (i.e. non-hardened) PowerPC processors are used in a voting scheme to greatly reduce the effects of single-event phenomena such as SEUs (see section A.1).

- *Resynchronization*: A combination of hardware and software logic (figure A.1 right) performs processor resynchronization in the event of a voting disagreement. A disagreeing processor is temporarily taken out of the voting scheme and recovered in under 1 ms. In addition, all voting processors are periodically scrubbed (or resynchronized) as a preliminary measure against buildups of SEUs. The refresh interval is not fixed—it is selected based on satellite orbit parameters and predicted SEU rates for a given orbit [127].

This successful design of the CALIPSO SBC based on the COTS 603r PowerPC has been upgraded with the next generation 7447A (SOI) G4-PowerPC processor running at 1 GHz and it can be readily deployed in future satellite missions [127].

A.3.3 The X2000 Avionics System

Another successful COTS-based fault-tolerant spacecraft computer design comes from NASA/JPL. JPL's goal in designing the X2000 is to achieve a flexible and scalable architecture, which can be reused in many future NASA missions and ultimately reduce the long-term costs of space explorations [130]. According to its designers, Chau et al., the X2000 is a "distributed, symmetric system of multiple computing nodes and device drivers that share a common redundant bus architecture" [133].

The X2000 avionics system depicted in figure A.2 shares many of the fault-tolerant architectural elements found on the CALIPSO SBC described in the previous section. The X2000, manufactured by British Aerospace Federal Systems, features the newer generation

Power PC 750 processor. Similar to the CALIPSO SBC, it employs industry standard mechanical and electrical interfaces as well as COTS serial bus architectures. Specifically Compact PCI (cPCI) is used for the local computer bus, while IEEE 1394 and I^2C are the system buses [130, 133].

IEEE 1394, a fast bus operating at 50-400 Mbit/s, is used for communicating spacecraft attitude, scientific data and other timing critical communication. I^2C is a slower bus operating at 100-400 kbit/s; it is used for communicating overall spacecraft health data collected by onboard sensors [132]. As shown in figure A.2, the buses connect multiple computing subsystems or nodes: flight computers, memory blocks, micro-controllers, optical communication units, sensor banks and scientific instruments.

The fault-tolerant features of the X2000 avionics system can be summarized in the following four layers [133]:

1. *Native Fault Detection*: The IEEE 1394 and I^2C buses already have some built-in fault detection. Invalid messages and non-responsiveness failure modes (see section A.3.1) are detected using bus capabilities such as cyclic redundancy checks (CRC) and acknowledgment.

2. *Enhanced Fault Containment*: The system's fault containment capacity is increased by an additional layer of hardware and software implemented on top of the native bus capabilities. This layer is designed to detect conflicts of node addresses on the two buses and babbling—failure modes which are more challenging. This layer also contains a baseline recovery mechanism for low-level faults on each bus [133].

3. *Mutually Assisted Fault Recovery*: On their own, the IEEE 1394 and I^2C buses have some shortcomings, but on the X2000 they are designed to aid each other in the isolation and recovery of difficult faults. For example, due to IEEE 1394's tree topology, a failed node or link can disrupt communication between the bus network sub-trees, making it hard to isolate faults. However, the I^2C bus can be used for backup communication; likewise, the IEEE 1394 bus can be used to assist the I^2C bus [133].

4. *Fault Control through System-Level Redundancy*: Both buses are fully physically redundant as illustrated in figure A.2 by parallel data lines. To save power, only one set of the buses is active at a time. However, when one bus fails, the backup set is powered up and used while the failed bus set is diagnosed and reset to a healthy state, possibly by eliminating a faulty link or node in the process [132, 133].

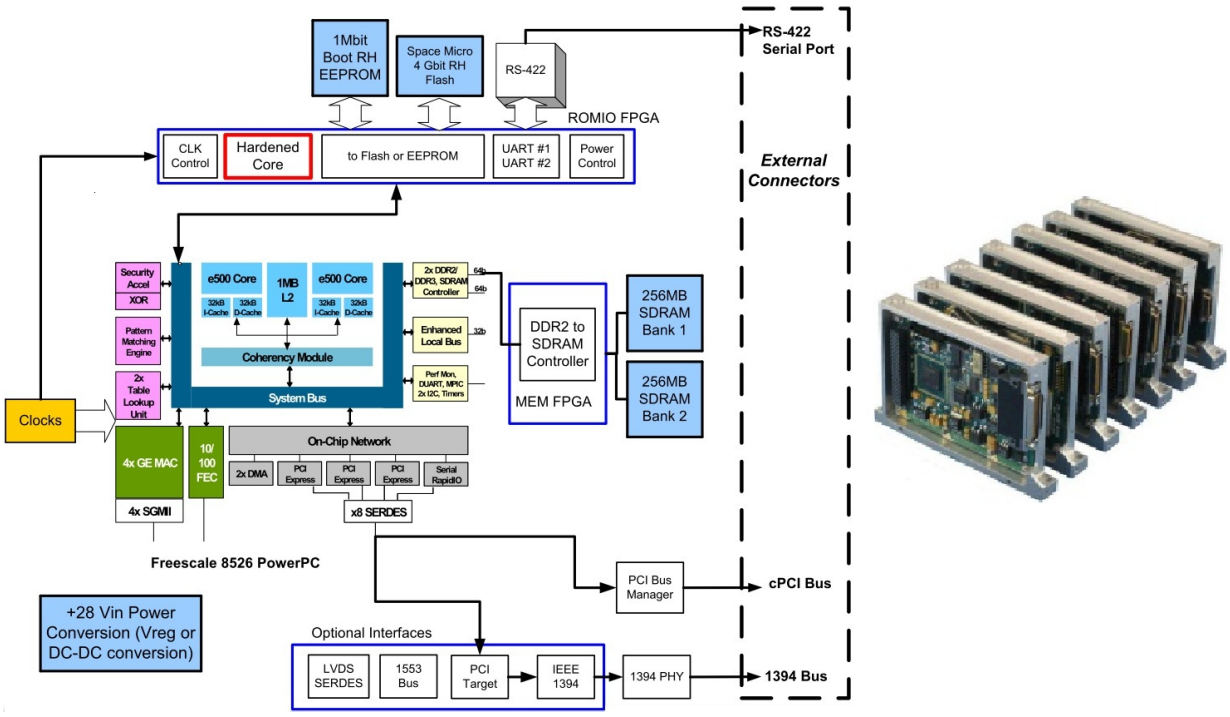

Figure A.3: Space Micro ProtonX-Box PowerPC SBC design and avionics suite cards [134].

The X2000 COTS-based architecture has been extensively evaluated. Quantitative models show that reliability levels of a 32-node instance of the system are more than 0.9999 at the end of an 11-year mission, compared to a reliability of 0.86 for similar-size non-fault tolerant systems [133].

A.4 Modern Commercial Payload Processing Hardware

OEM	Specifications	OS
Neptec	1.2 GHZ PPC, FPGA	Linux
IBM/BAE	RAD750 PowerPC	VxWorks
SpaceMicro	8-core PowerPC, FPGA, DSP	VxWorks
CPU Tech	PowerPC 440	Linux

Table A.2: Modern space-grade single-board computer manufacturers [122, 126, 134, 135].

State-of-the-art commercial spacecraft computing systems employ many of the fault-tolerant design elements discussed in the previous section. While certain critical compo-

nents are still radiation-hardened, many are COTS-based and feature cutting-edge hardware and software as summarized in table A.2. Most commercial payload processing systems today are comprised of multiple blades or slices (figure A.3 right) hosted in a compact chassis [128]. A great example is the REACT processor card by the Ottawa-based company Neptec Technologies. Their PowerPC-based card features a 1.2 GHz PowerPC MPC8548, 512 MB of DDR2 RAM with built-in error correction, as well as an on-board field-programmable gate array (FPGA) and an extensive list of connectivity features [122].

The most advanced space-qualified SBCs on the market today come from Space Micro [134]. Their Proton series SBCs offer high-end processing systems for a wide range of avionics applications. As depicted in figure A.3, Space Micro Proton designs employ top of the line PowerPC processors, and they also include FPGAs, digital signal processors (DSPs) and modern interconnects such as PCI express, SpaceWire and RapidIO [134].

A.5 Conclusion

Throughout its history, NASA has relied extensively on analog and digital electronics for all its space exploration programs. Even though payload processing systems have significantly evolved over the last 50 years, they still face the same challenges in the hostile environment of outer space. At first, all components of a spacecraft computer had to be fully radiation hardened and were a few technological generations behind. However, fault-tolerant COTS-based designs such as the CALIPSO SBC and the X2000 avionics system have shown that unhardened commercial computing hardware can be intelligently engineered to work reliably in space. The move to COTS components has led to dramatically improved spacecraft computing performance and an overall reduction of mission costs. Today, very high-performance COTS-based space-qualified SBCs can be readily purchased by commercial suppliers such as Neptec and Space Micro.