Trusted, Third-Party Authenticated, Quantum Key Distribution

by

Jonathan Hood

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 6, 2016

Keywords: quantum, cryptography, key distribution

Approved by

Drew Hamilton, Chair, Professor of Computer Science & Software Engineering
David Umphress, Professor of Computer Science & Software Engineering
Levent Yilmaz, Professor of Computer Science & Software Engineering
Joseph Sims, Director, Propulsion Engineering Center, Arctic Slope Technical Services

Abstract

This dissertation presents an algorithm that provides a way of establishing trust and authentication. The protocol negotiates a key using extensions to QKD algorithms that include non-repudiation and endpoint verification through a trusted third-party. The new algorithm proves the viability of implementing a trusted third-party in a QKD scheme.

Due to the capacity of quantum algorithms, the complexity of the new method is not meaningful to calculate using traditional big O methods. However, the Kolmogorov complexity calculation can be used to determine a form of the algorithm's complexity by measuring the operations it takes the algorithm to reach a successful state of entropy. Additional padding and negotiation with the third party yields a longer entropy calculation than QKD-only algorithms.

A reference implementation for the presented algorithm is provided. To test the reference implementation, a simulated quantum environment is created. The quantum simulation model and its correctness in implementing the newly created algorithm are validated for using standard model verification techniques.

Experimentation is set up as a "pass" or "fail" scenario. If any party is unable to unpad or decrypt a message, the algorithm is deemed a failure. If a party runs out of negotiated qubits, an entropy error is recorded and up to three retries are attempted. Experimentation on key sizes of at least 100 bits results in successful trusted key negotiation with 99.9999999987% confidence.

The results of the experiment culminate in a new algorithm, dubbed HHUYS16, which can be implemented using current technology. This could particularly be useful to government systems that require a quantum network and its assets to be secured. Implementation guidance is provided in the form of a QKD Security Implementation Technical Guideline (STIG); however, DoD implementation requires further coordination among organizations. Further improvements

and clarifications can be made with the National Institute of Standards and Technology's (NIST) proper identification of quantum-resistant encryption algorithms.

Acknowledgments

My wife, Caitlyn, and my children, Nathan and Gabriel, have suffered long hours without me so that I could finish this work. Thank you for your understanding and support!

Several people have endured the plethora of revisions to this dissertation, encouraging me and giving me constructive criticism along the way. These people include:

1. My wife, Caitlyn Hood
2. My parents, David and Sherry Hood
3. My advisor, Drew Hamilton

Thank you for not giving up on me and for steering me in the right direction!

Table of Contents

List of Figures

Chapter 1

Introduction

Security and privacy are basic human needs; we have a congenital, psychological urge to protect compromising information. From whispers on the playground to diaries locked beneath a pillow to an email account password, the need for tighter and more innovative security increases as we grow – individually and as a society. Quantum computing presents one new technological horizon whose security means are still developing alongside our knowledge of the quantum world.

Key-based cryptography has established itself as the most popular option for securing traffic through electronic communications channels. Keys are used in an encryption algorithm to protect the transmission of data on existing classical channels. Government networks in particular rely heavily on this field. Inherent in the design of any key-based cryptographic strategy is the need to share or distribute keys in a secure manner to other trusted parties. One of the most secure ways to achieve this distribution is by using Quantum Key Distribution (QKD).

1.1 Current Security Means – The Need for More Research

The need for cutting-edge security is particularly clear for the government and military whose secrets are a matter of national security. In 1997, the Department of Defense (DoD) issued DoDI 5200.40, establishing the Department of Defense Information Technology Security Certification and Accreditation Process (DITSCAP), a process for securing, validating, and authorizing components of national security systems (NSS).[29] By 2006, the DoD issued DoDI 8500.01 effectively transitioning DITSCAP to the Defense Information Assurance Certification and Accreditation Process (DIACAP).[32] In 2014, this process was once again reissued as DoDI 8510.01 into the Risk Management Framework (RMF).[33] Each of these frameworks describes

the certification and authorization policy to test components' cybersecurity posture within the DoD. RMF-accredited systems are categorized and given a baseline of controls from the National Institute of Standards and Technology Special Publication (NIST SP) 800-53 Rev 4.[73] Based on the categorization, a minimum of baseline controls is applied to the system, and further controls are then tailored and overlaid into that baseline as needed for the specific system.

Categorization of a system under RMF, particularly for National Security Systems (NSS), relies heavily on the CIA triad of cybersecurity. The CIA level of a system, when categorized using CNSSI 1253,[20] refers to a Low, Moderate, or High rating for each respective CIA category – its confidentiality (how important it is to remain private), its integrity (how important it is to remain unaltered), and its availability (how easily authorized entities can obtain access). For example, a tactical spy system behind enemy lines may be categorized as needing High confidentiality and integrity, but due to a constant need to hide behind enemy lines, may have Low availability. Such a system would be categorized as HHL on the CIA triad. Or suppose that a GPS satellite is sending out the time of day which several other systems synchronize with. Since the time of day is unclassified, common data, the confidentiality required for this transmission is Low, but the integrity (how accurate the time is) and availability of the system may be High. This system would be categorized as LHH.

While this process attempts to standardize an otherwise subjective evaluation of a system, new challenges arise in the area of quantum computing: how do current IA-enabled quantum computing assets align to the CIA triad? How should such systems be categorized and secured to be authorized for use on a government or commercial network? How can they be secured?

Several QKD solutions exist; each provides a reasonable solution for distributing a random encryption key between two endpoints, A and B, and giving a high guarantee of catching any third party who attempts to intercept that key. When the key is used in an authorized algorithm, both endpoints are assured that they are the only ones with the key. Due to this, existing QKD algorithms distribute a key that is highly confidential. However, confidentiality is just one part of

the CIA triad. What about integrity? Using the quantum and classical networks, how can point *A* know that her intended audience, *B*, is actually the one that negotiated the key with her?

Using a current standard algorithm for QKD, suppose that a malicious actor, *E*, stands in the way of *B* and completes the algorithm with *A*. *A* now thinks that she has negotiated a private key with *B*; however, she has really only negotiated the key with *E* who is impersonating *B*. *A* can be assured that her key is confidential between the two of them, but she has no assurance that the endpoint she is talking to is indeed *B*. This violates the integrity of the algorithm. A compromised integrity of the key can lead to a breakdown that severely and catastrophically affects the IA posture of the system.

1.2 Research Contribution

This dissertation concentrates on the Integrity portion of the CIA triad. The specific contributions to increase the Integrity guarantees include:

- Add endpoint authentication and non-repudiation guarantees to existing QKD algorithms. For example and walkthrough purposes, BB84 is used as the existing algorithm due to its simplicity (see Section 2.5.1).
- Formalize the endpoint authentication across a quantum network into an original algorithm, coined HHUYS16 (see Section 3.2)
- Formalize the requirements for authorizing such an algorithm on existing government networks (see Appendix C)
- Create a reference implementation using simulated data (see Section 3.3)
- Create a simulation framework to test the reference implementation (see Section 3.3.1)
- Outline the path forward for full DoD recognition and acceptance of authenticated QKD mechanisms (see Section 4.2)

If these changes are made, trusted, tird-party authenticated QKD can be achieved and managed.

Chapter 2

Survey of Literature

Exploring QKD requires an understanding of quantum physics – from its inception to its future. Quantum basics form the foundation for quantum experimentation which can be applied to the field of cryptography. Quantum Computing is a small piece of the field of Quantum Physics, and Quantum Cryptography is a small piece within the field of Quantum Computing. This chapter gives the history and theory that builds up to an even smaller subset of Quantum Cryptography: QKD. An understanding of the physics behind QKD is required for understanding the additional HHUYS16 process that builds upon it.

2.1 Quantum Basics

Before quantum theory can begin to apply to the computational realm, a few basic concepts must be covered. These concepts form the foundation for the theories and laws of a quantum-mechanically described universe that a quantum computing environment must abide by.

A "quanta" refers to a discreet amount, and Quantum Theory studies the properties of the smallest discreet physical amounts. In the quantum mechanical particle physics realm, there are elementary particles: the smallest particles known to science. These fundamental building blocks of the physical world can apply energy or have energy applied to them where the smallest unit of energy is defined by the particle's wavelength and Planck's Constant.[45]

Atoms were once thought to be indivisible elementary particles of the universe (the Greek "atoms" means "indivisible"). However, in 1874, physicist J. J. Thomson performed an experiment that showed the existence of subatomic units. In the 1960s, the Standard Model began to take form, defining what would become the properties and interactions of these elementary, fundamental particles.

Most quantum concepts are easiest to understand by examining their simplest experiments and gleaning the theories from the results of those experiments. The Double Slit Experiment and the Delayed-Choice Quantum Eraser experiment are the two essential areas of research for the quantum theory that deals with QKD. The theories and practice generated from these two experiments unlock underlying principles of QKD to explore – specifically, Heisenberg's Uncertainty Principle, Bell's Inequalities, the Copenhagen Interpretation, and Quantum Entanglement.

2.1.1 The Double-Slit Experiment

An introduction to quantum mechanics can be seen in the famous "Double Slit Experiment." A form of the experiment was first surmised by Thomas Young in 1802. [78] Then, in April 1925 at Bell Labs, Clinton Davisson and Lester Germer accidentally updated the experiment. [48] A form of the experiment is set around two control hypotheses.

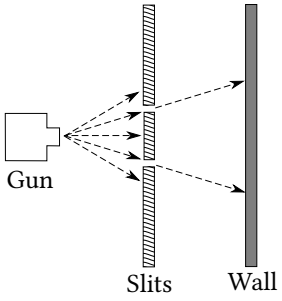

Figure 2.1: Double Slit Experiment: Gun

The first control hypothesis is explained by firing a "gun" from a fixed position randomly at a wall through two slits in a barrier (Figure 2.1). The slits allow the projectiles to hit the wall behind them, creating two areas at which the projectiles hit the wall. A "bullet" projectile can hit the wall by taking path A through the first slit or path B through the second slit. A bullet that hits the wall can only be described as having taken path A or B; a single bullet cannot take both paths.

The second control involves the travel of waves through the slits (Figure 2.2). The waves interfere with each other before hitting the wall. A wave can take path A through the first slit while simultaneously taking path B through the second slit; the waves from each path interfere

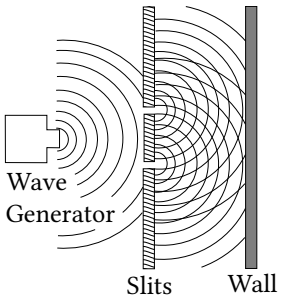

Figure 2.2: Double Slit Experiment: Waves

with each other so that they hit the wall in an interference pattern described as a combination of $A + B$.

Now we introduce the experiment at the quantum level using electrons instead of bullets:

Since a person cannot see electrons, suppose that an observer that is able to see or detect electrons is placed at slit A or slit B, and an electron gun is fired at the two slits. We can know which path the electron takes by determining if the observer has seen the particle. If the observer is placed at slit A and detects the electron passing by it, we know that the electron takes path A. If the electron is not observed, we know that it takes path B. The paths are deterministic. Indeed, when the experiment is run, it shows results as seen in Figure 2.1. But when the observer is removed, it is impossible to know which slit the electron is traveling through. The path is non-deterministic. The electrons hit the wall in the interference wave pattern from Figure 2.2! We can conclude from this experiment that quantum particles behave differently when they are being observed. Using the aforementioned example, we can clarify this conclusion by looking at the electron that took path B when an observer did not see it take path A. In this case, it would be more accurate to say that *knowledge* of a quanta's actions affect its behavior.

The experiment, however, has another twist. What happens if the observer is moved to the point on the backstop wall where the particle would hit if it took path A? If the observer is triggered, then the particle logically had to have taken path A. And if it is not triggered, then it would have had to have taken path B. When this experiment is run, the two slits as seen in Figure 2.1 are created! This leads to an interesting quandary: either the particle knew that it was going to be observed in the future (time $t + 1$) and took either path A or path B, or observing the

particle at time t changed its behavior in the past (time $t-1$). Either theory is interesting, and neither have yet to be proven.

2.1.2 Quantum Reality

Heisenberg's uncertainty principle defines the limits at which a matter wave's properties may be measured down to Planck's constant. An extension of this principle is seen in Bell's Inequalities in which a particle may be measured to determine if another observer has already measured physical properties of the particle. **Being able to measure a particle and determine if it has already been measured is key:** it means that we can know if data transmitted by the particle has been intercepted!

Heisenberg's Uncertainty Principle and the Copenhagen Interpretation

The Newtonian model of physics is deterministic. Heisenberg's Uncertainty Principle goes against that model, supposing that an accurate knowledge of a quantum particle's position and momentum cannot be established simultaneously. The Copenhagen interpretation holds that an unmeasured atom has no sense of reality: the act of measuring its attributes causes it to be realized in the very act of measurement. [52]

In 1935, Albert Einstein, Boris Podolsky, and Nathan Rosen (EPR or Einstein-Podolsky-Rosen) famously take the Copenhagen Interpretation apart, showing how the interpretation creates a paradox. According to EPR, two options for reality must exist: either "the quantum-mechanical description of reality given by the wave function is not complete," or "when the operators corresponding to two physical quantities do not commute, the two quantities cannot have simultaneous reality." [37] EPR then proves that if the first option is incorrect (and therefore the quantum theory is complete), then the second option is also incorrect (proving that the quantum theory is at best incomplete). Therefore, the nature of reality cannot be fully explained quantum-mechanically.

John Stewart Bell reexamines the EPR paradox in the creation of Bell's Theorem. Bell disproves the underlying assumption of the EPR paradox: local realism.[8] Bell starts with the same foundations as EPR: assuming fundamentally that the principles of reality and locality are true. Instead, his proof finds a method by which either the assumption of locality or the assumption of reality must be false.

Bell's Inequality and Quantum Entanglement

Generally, Quantum Entanglement states that the actions taken in one location simultaneously affect the knowledge of outcomes at a distant location. Suppose that two photons can have two states: if the first photon has state A, its entangled photon has state B. If the first photon has state C, then the second entangled photon has state D. The state of both photons can be described as either AB or CD. Quantum entanglement adds the property of superposition to these photons: a new state of $AB + CD$. [1]

An illustration of quantum entanglement can be described by using Bell's Inequality, a formula that culminates in a mathematical proof of Bell's Theorem.[8] To explain Bell's Inequality, it is best to posit a classical Newtonian physics problem, then adapt it to a quantum mechanically described model.

Suppose that two actors, A and B, agree to write two anticorrelated numbers on the first side of a sheet of paper: -1 and +1. One number is on the left end of the sheet of paper, and one number is on the right end (their order is randomly chosen). This process is repeated for the second side of the sheet of paper: -1 is written on either the left or right end of the paper (chosen at random), and its anticorrelated number +1 is written on the opposite end.

Let the left end of the first side be described as L_1 and the right end of the first side be described as R_1. Likewise, side two is divided between L_2 and R_2.

A possibility matrix for the setup of this experiment is as follows:

$L_1, R_1 \setminus L_2, R_2$	*-1, +1*	*+1, -1*
-1, +1	-1, +1, -1, +1	-1, +1, +1, -1
+1, -1	+1, -1, -1, +1	+1, -1, +1, -1

Figure 2.3: Bell's Inequality Decision Matrix in Newtonian Described Model

Now, suppose the sheet of paper is ripped in half so that A gets the left side and B receives the right side. If A looks at the first side of the sheet of paper and sees a -1, she will know that B's sheet has a +1 on the same side.

Bell's Inequality in this case is described as $M = L_1R_1 - L_1R_2 + L_2R_1 + L_2R_2$. To simplify, this can be written as $M = L_1(R_1 - R_2) + L_2(R_1 + R_2)$.

Using the simplified formula, we can create a range of possible values. R_1 and R_2, according to Figure 2.3, can either be $(R_1, R_2) = (-1, -1)$ OR $(-1, +1)$ OR $(+1, -1)$ OR $(+1, +1)$ where each possibility has an equal probability. In the case where R_1 and R_2 are the same, it is easy to see that the component of Bell's Inequality of $(R_1 - R_2)$ becomes zero. Likewise, when they are different, $(R_1 + R_2)$ becomes zero. When the possibilities in Figure 2.3 are plugged into the equation, it becomes trivial to see that the possibilities for the inequality can be described as $-2 <= M <= 2$. At least, those are the possibilities in a world described by Newtonian physics.

In a quantum mechanically described physics model, the numbers are not pre-written on each side of the page. Instead, reading a number would cause the anticorrelated number to instantaneously appear on the other party's half of paper by Heisenberg's Uncertainty Principle (Section 2.1.2). A's decision of which side of the piece of paper to read affects the value that B will read. Let the sides of the paper be described as reading a quanta in the orthogonal (0° and 90°) or non-orthogonal (45° and 135°) state. The possibilities, when factoring in the correlation of which "side of the paper" (ie: which basis to read the quanta in) becomes $-(2 * 2sin(45°)) <= M <= (2 * 2sin(45°))$.

2.1.3 The Delayed-Choice Quantum Eraser

To help understand why it is difficult to intercept quantum communication channels, Kim et al performed experiments with what they dubbed the "delayed 'choice' quantum eraser."[55]

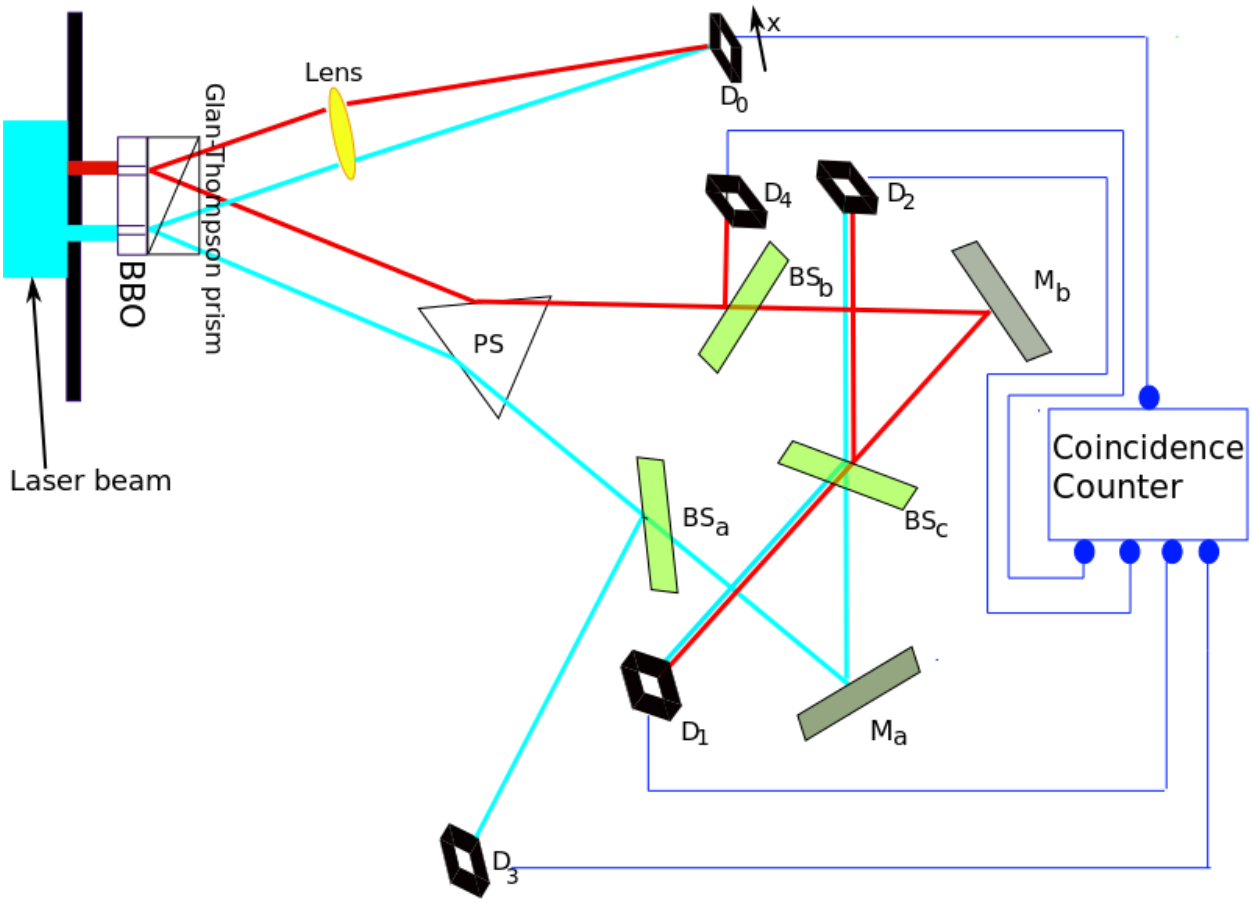

Figure 2.4: Delayed Choice Quantum Eraser

The delayed choice quantum eraser shows that actions performed to measure quanta in one location affect the behavior of quanta in another location. To illustrate this, follow the six possible paths a photon may take through the experiment:

- The photon would take the red path with 50% likelihood, at which point, it would:

  1. with 25% likelihood bounce off of Beam Separator BS_b and hit Detector D_4

2. with 12.5% likelihood pass through Beam Separator BS_b and bounce off of Beam Separator BS_c to hit Detector D_2
3. with 12.5% likelihood pass through Beam Separator BS_b and pass through Beam Separator BS_c to hit Detector D_1

- The photon would take the blue path with 50% likelihood, at which point it would:

1. with 25% likelihood bounce off of Beam Separator BS_a and hit Detector D_3
2. with 12.5% likelihood pass through Beam Separator BS_a and bounce off of Beam Separator BS_c to hit Detector D_1
3. with 12.5% likelihood pass through Beam Separator BS_a and pass through Beam Separator BS_c to hit Detector D_2

In summary, assuming an unbiased photon source, there is a 50% chance of the photon hitting Detector D_1 or Detector D_2, and there is a 50% chance of the photon hitting Detector D_3 or Detector D_4. The grouping of these detectors is significant because of one difference that was hinted upon in the Double Slit Experiment: when the photon hits either Detector D_3 or Detector D_4, the experiment reveals which path the photon took through the double slits at the beginning of the path. When it hits D_3, it took the blue path with 100% certainty. When it hits D_4, it took the red path with 100% certainty. The indeterminate path comes when the photon hits either Detector D_1 or Detector D_2: when this happens, the photon has a 50% probability of having taken the blue path, and a 50% probability of having taken the red path. It is impossible to know which one it took!

The interesting part of this experiment occurs at Detector D_0. When a photon takes the determinable path (triggering either Detector D_3 or Detector D_4), the pattern of photons landing on D_0 mimics the observed particle gun pattern of Figure 2.1. But when the indeterminable paths are taken (triggering either Detector D_1 or Detector D_2), the pattern on Detector D_0 mimics the wave-interference pattern of Figure 2.2!

The principle demonstrated here is that measuring an entangled photon at one point instantaneously affects the behavior of its entangled counterpart at a different location. For example, if an eavesdropper measured an entangled photon being sent from one entity to another, both sides of the communication can determine if an eavesdropper tried to gain access to their communication by determining if the photons are still entangled.

2.2 Quantum Experimentation

Quantum mechanically described experiments are still raw; as technology has increased, theories and postulations that have existed for decades are beginning to be realized. Some experiments have already been reviewed, such as the Double-Slit Experiment (see Section 2.1.1) and the Delayed-Choice Quantum Eraser (see Section 2.1.3). Additional current experimentation includes Quantum Annealing, implementations of Shor's Algorithm, Quantum Commutative Encryption (QCE), and QKD.

2.2.1 Quantum Annealing

Current implementations, such as the D-Wave quantum computers, use a process called Quantum Annealing to perform mathematical functions. This process is much slower than classical computing methods but allows the simulation of algorithms such as Shor's Algorithm (see Section 2.2.3). The largest publicized prime factorization on a D-Wave machine was the factorization of 56153.[61] The fact that quantum factorization based off of this method is slower than classical processors should not be underestimated: a D-Wave-enhanced computing environment cannot solve Shor's Algorithm, and it cannot factor a large number faster than classical algorithms at this time.

There is an effort by quantum computing companies to use prime factorization as a benchmark for their implementations. Researchers often claim to have broken the record for the largest number factored by a quantum computer (IE: "That quantum computation, [...] actually also

factored much larger numbers such as [...] 56153." [23]). These solutions use either quantum annealing or rely on adiabatic quantum computation.

2.2.2 Adiabatic Quantum Computation

Current implementations of Adiabatic Quantum Computation (AQC) utilize existing quantum annealing hardware to implement iterative version of a universal quantum computer.[67] It should be noted that no public research identifies AQC as having any notable performance benefits over classical computing algorithms.

2.2.3 Implementing Shor's Algorithm

Shor's 1996 Algorithm is the most well-known algorithm for breaking classical cryptography based on prime factorization. While the algorithm is public, useful implementation of it requires a Turing-complete quantum computer which is only a theoretical concept at this point. Nevertheless, current quantum computers, although incomplete, can be built and configured to implement a single prime factorization problem, and this has been accomplished with Shor's algorithm. While a Turing-complete quantum computer is necessary for building a *dynamically programmable* prime factorization algorithm, the remaining parts of Shor's algorithm can be implemented by statically configuring the circuitry to a single problem.

Most experiments using Shor's algorithm only use a few bits due to the incredible expense of current quantum computers. These are woefully short of solving for private keys of several thousand bits; therefore, their usefulness in modern computing environments is very low. The largest public implementation of Shor's algorithm uses 7 qubits and was built to factor the number 21.[58]

These experiments, while they do not show that a Turing-complete solution exists for all possible keyspaces, show that given enough money, a polynomial-complexity solution exists for each individual given key. The experiments show that Shor's algorithm works, even if it is prohibitively expensive to implement at this time.

2.2.4 Quantum Commutative Encryption

When dealing with Quantum Commutative Encryption (QCE), the data represented by a single unit encode for a 0 or 1 bit value. This is typically referred to as a "qubit" or "quantum bit" where a single qubit's information are contained in a single photon. For this section, "qubit" and "photon" can be used interchangeably.

Photon Orientation and Polarization

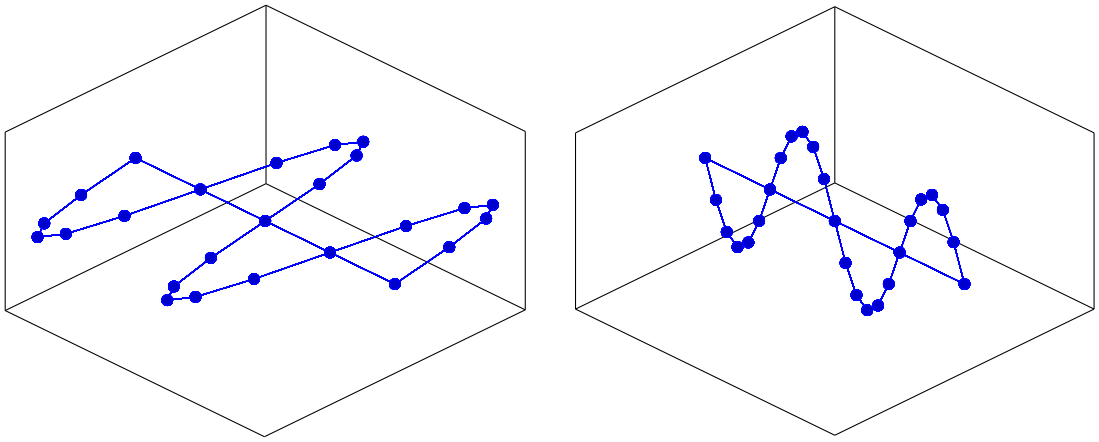

Figure 2.5: Horizontal and Vertical Polarization Bases

A photon can have a horizontal or a vertical polarization (see Figure 2.5). [54] Additionally, this polarization can be skewed by an angle, θ. Suppose that a photon is sent at either a 45° angle or a 0° angle. If a detector chooses to try to read the photon in the wrong orientation, he has a $sin(45°)$ of getting the right answer and a $\frac{1}{\sqrt{2}}$ chance of the photon coming through correctly in the first place. This means the detector will only have a $sin(45°) \times \frac{1}{\sqrt{2}}$ or 50% chance of getting the correct photon measurement. See Section 2.2.4 for how this can be used to establish a communication channel, forming the foundation of QKD.

Photon Spin and Qubit Encoding

After calibrating the receiver, the horizontal polarization of a photon as described in Section 2.2.4 can be referred to as a bit value of "1," and the vertical polarization of a photon can be

referred to with a value of "0." The polarization of a photon ($|\gamma\rangle$) can be described as horizontal ($|H\rangle$) or vertical ($|V\rangle$). [54]

The polarization of a photon is only half of the equation when considering communication using quantum entanglement; spin is the other half. The Faraday spin state of an electron can be made to interact with a photon. A particle can have $2n + 1$ spin states, so a spin-½ particle will have $2 * \frac{1}{2} + 1 = 2$ possible spin states. [4] A spin up state is shown as $|\uparrow\rangle$, and a spin down state is shown as $|\downarrow\rangle$. This gives four possibilities of polarization and spin of a photon interacting with the Faraday spin state of an electron:

1. a horizontally-polarized photon in the spin up state: $|H\rangle|\uparrow\rangle$
2. a horizontally-polarized photon in the spin down state: $|H\rangle|\downarrow\rangle$
3. a vertically-polarized photon in the spin up state: $|V\rangle|\uparrow\rangle$
4. a vertically-polarized photon in the spin down state: $|V\rangle|\downarrow\rangle$

Given two qubits entangled such that their states are $|01\rangle - |10\rangle$ (that is, one qubit is always 1 and the other is always 0), suppose that the evaluation of a qubit as 1 or 0 is performed by evaluating the spin of a photon around the vertical axis. If the first entangled qubit is measured, it can be discovered that it is in the spin up state: $|V\rangle|\uparrow\rangle$. This means that the other photon must be in the spin down state: $|V\rangle|\downarrow\rangle$.

There are competing theories about whether perturbation of the spin of the first photon will affect the spin of the second (Einstein's "spooky action at a distance"), but for the purpose of QKD, the more conservative hypothesis will be used. This hypothesis states that the spin of the first particle does not truly affect the spin of the second after the particles have been entangled (perturbation at one location cannot be detected at another). While at the moment of measuring the spin of one photon, the observer instantly knows the spin of the second entangled photon, in the absence of information about the measurements on the first photon, the second photon's spin state is unknown. [57]

While instantaneous communication may not be possible with quantum entanglement, the realities of entanglement are useful. Since measuring the photons changes their state, [49] this attribute can help determine if someone has observed data in-transit. The unknown spins of entangled, unmeasured photons can help create truly random numbers.

2.2.5 Quantum Key Distribution: An Introduction

QKD is a tested quantum process for setting up encryption by realizing the usefulness of realities such as entanglement. Truly random numbers are fundamental to QKD. As the focus of this dissertation, QKD will be discussed in more detail in Section 2.5.1. Now that the basics of the quantum world has been covered, there is another piece to QKD that must also be discussed: cryptography.

2.2.6 Quantum Entropy

Quantum entropy, for the purposes of this dissertation, refers to the modified Shannon Entropy of calculating a quantum coin flip. The entropy calculations for a quantum algorithm are meaningful in determining its complexity (see Section 3.4.1) and measuring the performance of simulated environments that implement the algorithm. The relative Shannon Entropy defines the "maximum compression possible when we encode into [similar] alphabet[s]."[71] In other words, the Shannon entropy is a measurement of the average amount of data that can be obtained from an event.[77]

Entropy calculations in QKD methods describe two important factors of a quantum algorithm: how much uncertainty is built up to perform the process and how much data is stored in each element of the quanta. The time it takes to build up entropy depends heavily on the quantum equipment being used. To account for the wide range of equipment, complexity calculations are usually written as a measurement of how much entropy must be built up to execute the algorithm.

The Shannon Entropy of a data source is described as $H(X) = -\sum_i p(x_i) \log p(x_i)$ where $p(x)$ refers to the statistical probability of receiving message x.[70] The Shannon Entropy can then be used to determine the average number of qubits that must be sent from the source in order to encode the expected amount of information.[12]

Given a standard coin flip, the Shannon Entropy of agreeing on a single flip would be $H(X) = -0.5 \log_2 0.5 = 0.5$. This is what we would expect: if information can only be gleaned from the entropy source when an observer guesses a coin flip, each flip will average 0.5 Shannons of entropy.

The quantum coin toss involves an extra layer of uncertainty: the act of choosing a measurement basis changes the Shannon Entropy calculation, and the calculation is dependent upon the measurement bases in the "alphabet." In the example of a two-state protocol such as BB84 or E91, the Shannon Entropy calculation changes because each endpoint must choose a measurement basis. The probability distribution becomes $\frac{1+sin(45^\circ)}{2}$. The Shannon Entropy for a quantum coin flip then becomes $H(X) = -\frac{1+sin(45^\circ)}{2} \log_2 \frac{1+sin(45^\circ)}{2} = 0.195$. Likewise, if measurement options are available as per the SSP-based algorithms, the probability becomes $\frac{1+sin(30^\circ)+sin(60^\circ)}{3}$ and the Shannon Entropy becomes $H(X) = \frac{1+sin(30^\circ)+sin(60^\circ)}{3} \log_2 \frac{1+sin(30^\circ)+sin(60^\circ)}{3} = 0.270$.

The Shannon Entropy (and by extension, the Vonn Neumann entropy) play a critical role in the measurement of Bell's Inequalities.[75] But another important role of the entropy calculations is to determine the number of qubits that should be sent to stage the algorithm. There enlies the problem of quantum entropy: by relying on uncertainty, quantum algorithms must rely on their platforms to build up enough entropy to perform the algorithm. If the platform does not provide enough entropy (in the form of negotiated qubits for QKD), an entropy failure prevents the algorithm from succeeding.

When enough qubits are negotiated to perform the algorithm, the system achieves a successful state of entropy. It is important to realize that a failure in entropy does not mean that the algorithm cannot be accepted as a valid algorithm. In the same way that cutting the ethernet cord to a node in the middle of a TCP session does not invalidate the algorithms that make up

TCP, a failure in the underlying platform of QKD does not necessitate the rejection of an entire algorithm.

2.3 Cryptography

A discussion of current cryptographic methods is needed to understand their shortcomings in a quantum realized environment. Current, trusted, classical algorithms are discussed in this section along with possible methods for breaking them in a quantum environment. It is important to realize: each of the quantum methods used to break each of these classical cryptography methods has been shown to work experimentally for very small key sizes. As quantum computing environments are able to perform calculations on larger bit sizes, the danger of a quantum environment being able to break modern cryptographic key sizes increases.

2.3.1 Current Cryptography

Being able to measure Bell's Inequalities and detect if a communication has been compromised is a very exciting proposition, but it still must demonstrate that it is better than current cryptographic methods in some way. Some of the more popular algorithms are analyzed here from a security perspective.

Three of the most common forms of encryption are provided. Each of these cryptography methods have implementations, depending on their keysize, that are currently NSA-approved for encryption of data that are Top Secret. There is no computationally easy way to solve these algorithms using raw computing power, and to bruteforce or solve them would likely require so much time that the usefulness of the encrypted information once it is decrypted may be diminished.

The realm of quantum physics provides a new type of calculation tool: one in which the states of a problem can be calculated in an infinite number of ways simultaneously. Consider again the Double-Slit Experiment (Section 2.1.1). In it, the measured quanta took a path through both slits which were described as $A + B$. The theory is that the quanta took both slit A and slit B, then interfered with itself until it hit the backstop. Instead, quantum theory proposes that the

path taken by the quanta is entirely A and entirely B, but is only described as $A + B$ when it is observed by the backstop. Prior to being observed, the quanta was actually in all of the states described by the wave function of wave A and wave B ($A + B$). Measurement of the state resulted in a single result and broke the quanta's infinite possible states instantaneously.

Being able to have a quanta that acts in infinite ways forms the basis for measuring the result of those actions. When those infinite actions allow the measurement of the quanta to reveal information about an exponentially difficult mathematical problem, a quantum algorithm has the potential of solving the exponential algorithm instantaneously or in quantum polynomial time.

RSA Encryption

Ron Rivest, Adi Shamir, and Leonard Adleman theorized an algorithm using what is known as a trapdoor function. A trapdoor function is one that is easy to solve, but only for the holder of special variables. RSA uses prime factors as inputs to its trapdoor function. Knowing the prime numbers (which form the private key) allow the holder of a private key to perform the decryption function quickly. An attacker must calculate the prime numbers corresponding to the public key in order to figure out the private key and decode the message. But factoring a large number is very difficult for a computer; the best algorithms are currently Pollard's algorithm and the General Number Field Sieve (GNFS).

Pollard's algorithm can reduce the key space when factoring prime numbers by nearly a factor of 4. This means, for a 2^{1024} number, the keyspace to check for factorization is reduced to 2^{1022} possibilities for large keyspaces. [63]

In Riesel's GNFS, based off of Pollard's original Number Field Sieve, Riesel optimizes the GNFS to create an algorithm that should only be used for very large prime factors. As its keyspace is still asymptotic, the algorithm has the potential of exceeding a bruteforce keyspace for very small inputs while reducing the keyspace to close to polynomial time as the input size approaches infinity.[65] Figure 2.6 describes the algorithmic complexity for inputs greater than 100 bits.

$$O\left(\exp\left(\left(\tfrac{32}{9}b\right)^{\frac{1}{3}}(\log b)^{\frac{2}{3}}\right)\right)$$

Figure 2.6: Riesel's Optimized General Number Field Sieve

The difficulty in calculating the secret key comes from limits on computing power: an RSA-solving classical computing algorithm is prohibitively expensive when utilizing modern computing technology. To be usable on limited resources, the prime factorization must be performed in at most polynomial time. Such an algorithm exists in the quantum realm: Shor's algorithm.

Shor's algorithm (see Section 2.2.3) is a polynomial quantum algorithm that requires a Turing-complete quantum computer to run.[72] While a Turing-complete quantum computer is prohibitively expensive and difficult to build currently, Shor's algorithm has been *publicly* tested to successfully factor numbers up to 7 bits in size in polynomial time; these initial tests of Shor's algorithm show its promise should larger quantum computers be made.

Using Shor's (or another quantum phase measuring) algorithm, trapdoor functions based off of prime numbers become useless as the private key can easily be solved.

Discrete Logarithms

Discrete Logarithms, like Prime Number Cryptography using a trapdoor method, use a function that is very difficult for a computer to solve. Since there are no unknown variables, discrete logarithms are not trapdoor functions; they are just difficult to solve in one direction.

Classical algorithms for solving discrete logarithms exist that can operate exponentially in $O(\sqrt{n})$ keyspace, but this is still in the realm of exponential algorithms. Additionally, if a prime factor is known, Pollard's Rho algorithm can operate exponentially in $O(\sqrt{p})$ where p is the prime factor.

Shor's 1994 algorithm adaptation for solving prime number factorization was optimized and modified in 1996 to apply to discrete logarithms using the same general method of simultaneously measuring all quantum phases between the limits of the solution space to obtain a quantum measurement (and a cryptographic solution) for discrete logarithms (see Section 2.2.3).[72]

Elliptic Curves

Elliptic curve cryptography is usually implemented by taking the best of the concepts of prime number factorization and discrete logarithms. In it, an elliptic curve function is created, based on the finding of a very large prime number. After defining a finite field, the curve along that field has points satisfying the original equation. These points, usually calculated separately by two parties, form the basis of creating a shared private key that can be calculated by both parties without sharing much data.

In 2003, Proos took Shor's Discrete Logarithm algorithm and applied it to the Elliptic Curve Discrete Algorithm, proving a viable quantum computing attack against classical elliptic curve cryptography.[64] Proos uses the simplification of Shor's algorithm provided by Griffiths and Niu to solve the prime factorization and discrete logarithm algorithms of the Elliptic Curve, then define quantum solutions to each of the transformation functions defined that are performed against the curve's finite group space.[50]

2.3.2 Fixing Cryptography with Quanta

Fortunately, quanta also provide solutions to the cryptography they break. By measuring for Bell's Inequality in quanta, endpoints of communication can obtain a high degree of certainty of whether or not a message has been compromised by an eavesdropper. Measuring for Bell's Inequalities therefore guarantees a high level of Confidentiality in the CIA levels. The very nature of Heisenberg's Uncertainty Principle (Section 2.1.2) also creates non-deterministic random numbers.

If a quantum channel can provide assurance that information is not eavesdropped upon, and the nature of the checks produce non-deterministic random numbers, then quantum communications channels can be used to generate and distribute encryption keys between endpoints. Furthermore, the keys can be thrown away if an eavesdropper is detected.

2.4 Quantum Cryptography

QKD is a small piece of the overall study of quantum-safe cryptography. Quantum cryptography is the study of cryptographic methods that are not easily solvable using a Turing-complete quantum computer. With algorithms such as Shor's Algorithm (see Section 2.2.3), a Turing-complete quantum computer is capable of using qubits that determine their state based on the measurement of a quantum particle - a particle that has infinite measurement possibilities. A Turing-complete quantum computer is capable of making all those measurements simultaneously.

Algorithms susceptible to infinite quantum measurement attacks include the ones listed in Section 2.3.1. It may seem that if a quantum algorithm can solve for infinite solutions simultaneously, it can perform a bruteforce attack on nearly any algorithm. *Such an assumption would be largely, though not completely, correct.* Consider, for example, a bruteforce attack on a one-time pad. Bruteforcing the cyphertext of a one-time pad results in many possible plaintext solutions; "We attack at dawn" has just as much likelihood as being the plaintext of a one-time pad as "we attack at noon," "we attack at 1400," or "let's kill Hitler." In contrast, the prime factorization problems have a distinct solution.

When an algorithm is not susceptible to a quantum attack, it is described as "quantum-safe."

2.4.1 Quantum-Safe Encryption Algorithms

As already mentioned, the Vernam One-Time Pad is a quantum-safe encryption algorithm as a bruteforce attack on the data encrypted by the algorithm yields no useful information. It is the easiest algorithm to understand and will form the basis of the quantum encryption algorithm that is used in Section 2.5.1; however, it could be replaced with other algorithms that are also deemed to be quantum-safe.

Lamport Signatures

The natural progression from a Vernam One-Time Pad to a slightly more complex algorithm goes through a class of algorithms that utilize "Lamport Signatures." Lamport Signatures bring the public/private key paradigm to Vernam One-Time Pads. A Lamport Signature, in its simplest form, consists of a one-way function, f. This function, which could be a hashing function, a one-time padding function, or really any non-deterministic one-way function is used for digital signatures. The signer generates two random secret strings, S_{k0} and S_{k1}, such that the public key of the message to sign is defined as $f(S_{k0})$ when the bit to sign is 0, and $f(S_{k1})$ when the bit to sign is 1. Distributing the public key immediately releases information about the secret values, but does not give an attacker enough information to forge signatures. As such, this is a one-time signature algorithm.[62] Nevertheless, the keys that result from the process can be used to sign a message can be used in place of a simple padding function.

Lamport Signatures, by design, are identical cryptanalytically to one-time padding functions. Their main applicability in the algorithm provided in Section 3.2 comes in the form of allowing Trent to more securely sign the authentication messages to Alice and Bob. This increases the security and complexity of the resulting algorithm.

Lattice-Based Signatures

NTRUSign, the digital signature algorithm that stems from NTRU encryption (see Section 2.4.1), applies the concepts of a Lamport signature (see Section 2.4.1) using two definitions across a lattice instead of two purely random numbers. Each time a signature is made, data about the private key is leaked; however, current algorithms estimate that thousands of signatures are needed to perform an attack against the private key.[36]

Since the data leakage inherent in this algorithm is not currently known to pose the release of a private key, it can be used for a one-or-two-time signature in the algorithm defined in Section 3.2 to reduce the complexity of the signing portion of Trent's communication.

Lattice-Based Cryptography

Lattice-based cryptography has gained in popularity over the past few years, primarily with the release of NTRU-compliant, open-source, cryptography libraries.

NTRU, created in 1996,[2] is a specific implementation of lattice-based cryptography. Given a finite lattice, there are two mathematically difficult vector functions in relation to a lattice: finding the shortest vector or finding the closest vector in relation to a variable in addition to the lattice definition passed as a parameter to the NTRU algorithms. Open-source NTRU algorithms generally opt for the shortest vector option due to their ease of implementation, low resource consumption, and lack of a known algorithm that can break it. The shortest vector problem is not believed to be NP-hard, but the current algorithms to solve it are exponential in nature. The downside to using the shortest vector option is that there is also no mathematical proof of its security.

In spite of its lack of mathematical proofs, NTRU encryption based off of the shortest vector problem has been analyzed for nearly two decades. While some algorithms, including quantum simplifications, exist for reducing a finite lattice field, the algorithm has no classical nor quantum algorithm that is known to break it. In RSA encryption, Shor's algorithm takes advantage of the nature of finding prime factors by assuming that a given input is the product of two such numbers. Plotting these numbers on a sine wave and measuring for their possibilities along the wave allows Shor's algorithm to solve for the factors. Proos' algorithm (and some modifications to Grover's algorithm) breaks elliptic curve cryptography and other transformation algorithms by taking advantage of the Fourier transform mechanics of elliptic curve distributions. Since lattices do not have such transformations (the transformations are the actual encryption coefficients), no current quantum algorithm can solve lattice-based cryptography faster than a classical computing algorithm.

It is important to remember these facts about NTRU and other lattice based encryption schemes:

1. There is no mathematical proof for NTRU.

2. The current lattice-based cryptographic methods that do have a proof do not have efficient implementations.[44]

3. There is no known quantum algorithm for solving a lattice.

4. There are quantum algorithms for reducing a finite algorithm used by NTRU encryption,[42] and, when combined with classical lattice solving algorithms, they can reduce the keyspace to search when solving for a key. IE: The encryption scheme is not as secure mathematically as a one-time pad.

Lattice-based cryptography may or may not be the next step in cryptographic algorithms, but it does show that random details on ideal lattice distributions can keep transformation-solving quantum algorithms from easily breaking the encryption schemes.

The two provable classes of lattice-based cryptography require an infinite lattice or, as defined by Gentry, ideal lattices or cosets thereof.[44]

2.5 Quantum Key Distribution

The experimental algorithms in this section utilize mostly current technology and can be implemented in quantum environments. In fact, DARPA's quantum network used BB84 in its original implementation, and due to the simplicity of BB84, it is chosen as the underlying QKD algorithm for HHUYS16.

2.5.1 Foundations and Current QKD Algorithms

Each basic QKD algorithm contributes a new method or thought experiment into the quantum realm. While the existing algorithms can be classified as either classical, uncertainty-based, prepare-and-send algorithms or as entanglement-based EPR algorithms. The contributions and methodologies of several of the most popular QKD schemes are examined, and each method has been considered for implementation into HHUYS16 (Section 3.2). The attributes that lead

to changes in HHUYS16 to implement each of these algorithms (note that BB84 and E91 are the basis for the reference implementation) are given extra consideration in this research.

Uncertainty	*Entanglement*
BB84	E91
B92	BBM92
4+2	
GV95	
KI97	
SSP99	SSP99/EHHPK02
C00	
SARG04	SARG04
	TL06

Figure 2.7: Current Algorithms for Quantum Key Distribution

Algorithms for QKD are usually divided into two types. Photons can be read directly from a sender to determine the key (see Section 2.2.4), or the reading of quantum-entangled qubits can be used (see Section 2.2.4). Using directly sent photons to distribute keys is theorized by Charles Bennett and Gilles Brassard (culminating in the BB84 protocol in Section 2.5.1). The use of entangled photons builds the basis for Artur Ekert's protocol, E91 (see Section 2.5.1). Figure 2.7 shows which algorithms are based on Heisenberg's Uncertainty Principle (Section 2.1.2) and which are based on Quantum Entanglement (Section 2.1.2).

Nearly every protocol based off of BB84 (which include all Heisenberg Uncertainty based algorithms listed here) can be converted to Entanglement based algorithms. By changing the source from a send-and-receive origin to entangled photons, BB84 basically becomes the E91 protocol. [11]

BB84

Suppose that sender A wants to communicate with receiver B and has both an open-space traditional communication channel and a quantum channel. A can send a photon in four possible ways. First, A chooses a basis to send a bit in: the rectilinear basis ($\oplus$) or diagonal basis ($\otimes$). When sending in the rectilinear basis, A can encode a bit 0 value by sending a horizontally polarized

photon (0°) or a bit 1 value by sending a vertically polarized photon (90°). In the diagonal basis, a 45° photon represents a 0 value while a 135° photon represents a 1 value. *A* performs two random steps: choosing a basis and choosing a value.

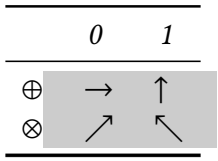

Figure 2.8: Possible Photon Orientations

A is the only entity that knows what basis and what polarization each bit was sent in. This is where some properties of the quantum channel become useful. Suppose that *A* sends a 0 bit value in the diagonal basis (45°). *B* then has a choice to read the bit and must choose a basis to read it in. If *B* correctly chooses the diagonal basis, a 0 value is read. If *B* incorrectly chooses the rectilinear basis, a 0 is returned with probability $sin^2(\Theta)$ and a 1 with probability $cos^2(\Theta)$. Given the 45° angle the photon was sent in, both probabilities are $(\frac{1}{\sqrt{2}})^2$ or 50%. *B* has a 50% chance of guessing the right basis and a 50% chance of getting the correct bit randomly despite picking the wrong basis. In total, *B* has a 75% chance of getting the correct bit.

After *B* receives the bits, *A* can broadcast on the open traditional channel which basis was used to send each bit, and *B* can compare them to the basis that was chosen randomly. On average, *B* will have picked the correct basis 50% of the time. *B* can then report back which bits the correct guess was made for, and those bits will be used to generate the private key. [10] The shared, private key was never sent across an open channel; therefore, it can be used to compare Bell inequalities to check for eavesdroppers whom have looked at the key. The key also has the potential of securely being changed as often as needed.

Many more bits are sent than needed to compensate for 50% of the bits being thrown away. Additionally, after the bits corresponding to the incorrectly chosen basis are thrown away, some of the correctly chosen basis bits are compared to verify the private key without sending it across the traditional channel. [39]

Quantum Transmission															
A's random bits	0	1	1	0	1	1	0	0	1	0	1	1	0	0	1
A's random bases	D	R	D	R	R	R	R	R	D	D	R	D	D	D	R
A's photon	↗	↑	↖	→	↑	↑	→	→	↖	↗	↑	↖	↗	↗	↑
B's random bases	R	D	D	R	R	D	D	R	D	R	D	D	D	D	R
B's received bits	?	x	1	x	1	?	?	0	x	?	?	1	x	0	1
Public Discussion															
B reports bases	R		D		R	D	D	R		R	D	D		D	R
A reports correctness			✓		✓			✓				✓		✓	✓
Shared Bits			1		1			0				1		0	1
B reveals random bits			1											0	
A confirms random bits			✓											✓	
Outcome															
Remaining shared secret					1			0				1			1

Figure 2.9: Illustration of BB84

Bennett and Brassard give the illustration shown in Figure 2.9 to help explain the protocol. When *B* receives a bit of "x", that means the photon could not be read due to equipment error or corruption. When *B* receives a bit of "?", the bit is randomly 1 or 0.

E91

Ekert's algorithm is essentially the same as BB84 with the additions of quantum entanglement and a different mathematical proof. A photon sent by *A* corresponds to azimuthal angles $\phi_1^a = 0$, $\phi_2^a = \frac{1}{4}\pi$, $\phi_3^a = \frac{1}{2}\pi$ and $\phi_1^b = \frac{1}{4}\pi$, $\phi_2^b = \frac{1}{2}\pi$, $\phi_3^b = \frac{3}{4}\pi$. Superscripts "A" and "B" correspond to the orientation of the analyzers used by sender *A* and receiver *B* along the vertical axis. [38]

If Bell's Inequality (see Section 2.1.2) holds, there may be an eavesdropper present! [46] After a similar "Public Discussion" phase to BB84's, E91 tests the orientations of some bits to verify that Bell's theorem holds (and thereby confirming if an eavesdropper is present).

B92

In B92, Bennett explores different ways of sending photons to achieve secure key distribution. In his exposition, he gives both an EPR and non-EPR version of the algorithm. The EPR

version is expounded upon in BBM92 even though it is defined by Bennett in B92. In the non-EPR version, A performs the following steps: [9]

1. A sends a random sequence of photons polarized as horizontal ($\leftrightarrow$), vertical ($\updownarrow$), right-circular ($\circlearrowright$), or left-circular ($\circlearrowleft$).

2. B chooses random bases to measure the photons.

3. A and B negotiate publicly about which bases were correctly chosen, similar to BB84.

4. Data are interpreted in binary as $\leftrightarrow = \circlearrowleft = 0$ and $\updownarrow = \circlearrowright = 1$.

5. A random subset of the key are chosen for verification.

6. QoS safeguards are introduced by attempting to correct the key. If a checked bit is wrong, it indicates either an eavesdropper or an off-by-one sending error. To fix the off-by-one error, a random bit is thrown away between the check bits to see if the keys then matched. Probability in detecting an off-by-one error depends on how many checked bits there are. Given k checked bits, the probability of detecting an off-by-one error is $1 - 2^{-k}$.

BBM92

B92 also works as an EPR-based algorithm. The difference in the algorithm is how the photons are prepared. In EPR schemes, A and B both do not know the nature of each of their entangled photons. Therefore, both A and B pick a random base to measure in: either the rectilinear (+) or circular ($\bigcirc$) base. These bases are then negotiated by both parties after measurement. [9]

BBM92 then proves that any non-EPR protocol can be converted to an EPR entanglement protocol. BB84 is used as the primary example, and the changes of B92 are included when converting to an entanglement protocol. [11]

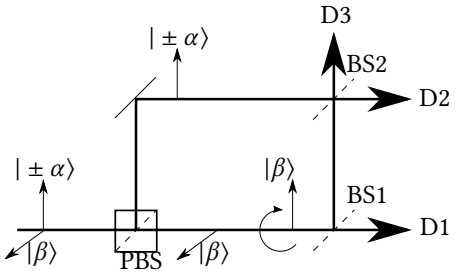

Figure 2.10: Receiver of 4+2 Protocol

4+2 Scheme

The 4+2 protocol uses the traditional four photon send states ($\rightarrow$, $\nearrow$, $\uparrow$, and $\nwarrow$) and adds to it an additional two possible send states. In this protocol, the classical four polarizations are sent by a weak state identified by $|\pm\alpha\rangle$, and the two additional polarizations are sent on an orthogonal strong channel identified by $|\beta\rangle$. Huttner, Imoto, Gisin, and Mor illustrate in Figure 2.10 how to use a polarization beam splitter (PBS) to separate these states. After separation at the receiving end, $|\beta\rangle$ is passed through a mostly transmitting beam splitter (BS1) that sends a portion of $|\beta\rangle$ equal to the amplitude of $|\alpha\rangle$ to interfere with $|\pm\alpha\rangle$ while the majority of $|\beta\rangle$ is sent to detector 1 (D1). [53] The interference is then detected by D2 and D3.

GV95

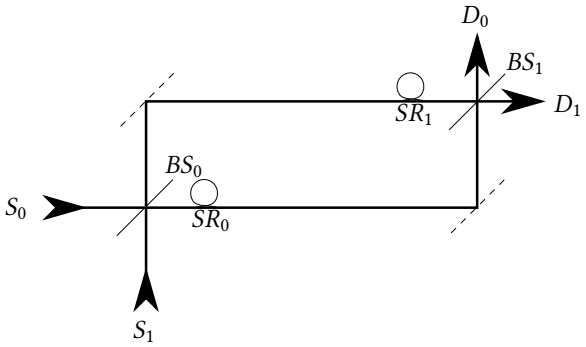

Figure 2.11: The Goldenberg/Vaidman Scheme

The Goldenberg/Vaidman Scheme introduces an eavesdrop detection scheme on orthogonal quantum states. In Figure 2.11, the sender chooses whether to send a 0 (using S_0) or a 1 (using S_1). The beam is split into two wavepackets taking route $|a\rangle$ along the top channel and $|b\rangle$ along the bottom channel. By using a Mach-Zehnder interferometer [76] in circuit with storage rings

SR_0 and SR_1, bits that emerge at D_0 indicate a 0 bit, and bits that activate the D_1 detector indicate a 1 bit. [47]

By introducing delays into the paths, an eavesdropper cannot access data in the top path before accessing data in the bottom. By doing so, the eavesdropper would be revealed when the delay on the receiving end is triggered at the wrong time.

KI97

The Koashi/Imoto Scheme is identical to the Goldenberg/Vaidman Scheme with the addition of a π rotation adjustment on the lower path. [56] This phase adjustment provides better error correction than the Goldenberg/Vaidman Scheme. Additionally, an eavesdropper is more likely to be detected than with the previous protocols.

SSP99: Prepare and Measure

The Six-State Protocol adds an extra base to the existing BB84 protocol. [7] The three bases that make up this protocol are akin to measuring a photon along the x, y, and z axis. [15] During the negotiation, the axis along which the photon was sent is confirmed by both parties. By adding extra dimensions, security is increased. [16]

C00

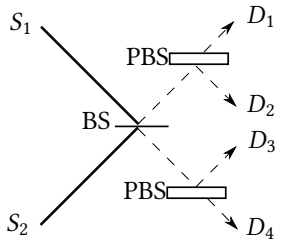

Figure 2.12: The Cabello Scheme

Cabello improves on the KI97 scheme by adding a special analyzer to B's receiving logic. Figure 2.12 shows how the sending bits can be demultiplexed into their four states to trigger each of the destination sensors. [17]

SSP99 (aka EHHPK02): Entanglement-Based

As with other EPR entanglement based algorithms, the setup for EHHPK02 is identical to SSP99. A source of an entangled photon sends to both the points A and B who must choose one of the three bases to read the bit in. On average, both endpoints will randomly pick the correct base $\frac{1}{3}$ of the time. This does require sending more bits to create a secure cryptographic key. [40]

SARG04: Prepare and Measure

Scarani, Acín, Ribordy, and Gisin's protocol begins identically to BB84 in that sender A chooses one of the two bases and one of the two angles to send each photon in. The difference comes in the sifting phase: instead of announcing which basis each photon was sent in, the non-orthogonal state is announced. If B guessed the correct non-orthogonal state and polarization, the bit is kept. The downside is that B will only guess correctly 25% of the time (compared to 50% with BB84); however, there is a greater guarantee of security when using weak photon sources (such as weak laser pulses). [69]

SARG04: Entanglement-Based and TL06

Tamaki and Lo start with SARG04 and create two new protocols with it: one in which entanglement is used (an extension to SARG04) and a combination of SARG04 with SSP99 (in the form of EHHPK02). By adding another base, TL06 makes SARG04 more secure. [74] [14]

2.6 Summary

The quantum-mechanically described universe and its applications to computational environments have taken long, specific strides over the past two decades. In the past four years, implementations of Shor's algorithm have been realized on a small scale, and speedups using quantum technologies are not far behind. Security is at a tipping point: these algorithms are real. They will break current cryptographic methods. The good news is that cryptographic methods exist for which quantum computers do not provide any advantage in breaking. Additionally,

the quantum world does not merely break current cryptographic methods; it provides new and unique solutions in the realm of cryptography. Primarily, QKD is a real and current method for distributing encryption keys – keys that can be used in a quantum-resistant algorithm – in a safe and reliable manner between endpoints. Nevertheless, more research is required to authenticate these endpoints using a quantum computing environment.

Chapter 3

Contributions and Experiment

After surveying the literature, no trusted, third-party algorithm was found that could authenticate both endpoints and provide non-repudiation guarantees. This dissertation proposes a solution to that problem in the form of HHUYS16 – an algorithm capable of establishing the integrity guarantees missing in current literature. HHUYS16 introduces T, a new party in the Quantum Key Distribution scheme, which can authenticate the endpoints in Quantum Key Distribution.

3.1 Experiment Setup

The primary contributions of this dissertation come in the form of a new experiment and an algorithm that satisfies the requirements of the experiment: the successful exchange of a private key between two parties that can trust each other without the experience of any of the failure criteria.

The experiment is set up by having three entities: A (Alice), B (Bob), and T (Trent). HHUYS16 assumes that registration has occurred with T (example registration scenarios are provided), and that both A and B are unauthenticated with one another. They must use their trust in T to authenticate with one another and negotiate a key.

3.1.1 Hypothesis

Hypothesis: An algorithm exists that establishes endpoint verification through a trusted, third-party entity without revealing encryption algorithm details or leaking information about the endpoints or their data.

Since no algorithm currently exists that defines a trusted, third-party guarantee to existing quantum networks, the experiment is set up as a "pass" or "fail" scenario. Extensions to the experiment that include optimizing performance and entropy calculations are detailed in Section 4.1.

3.1.2 Success Criteria

The successful exchange of a private key between A and B without any of the failure or indeterminable criteria occurring indicates a successful simulation of the algorithm.

3.1.3 Failure Criteria

Several criteria can cause a failure of the algorithm:

- T is able to determine the negotiated key between A and B
- There is not enough entropy to pad any of the activity on the classical channel
- Any of the temporary keys are compromised
- A or B fail to negotiate their measurement bases for the original key
- A or B fail to verify its private key with T, indicating a compromised trusted authority
- T fails to verify the private key of A or B, indicating attempted impersonation or compromise of that node
- The technical failure of any of the steps in the algorithm

In the event of a failure, the simulation is programmed to throw an exception with a message about what stage the exception occurred in.

An additional check in the simulation after the algorithm completes violates the quantum environment separations but is necessary for verifying the correctness of the algorithm. The original message from A to be encrypted using the negotiated key is compared to the decrypted

message received by B to verify that the messages are identical. A and B, by design of the algorithm, are not to communicate this data with each other without first encrypting it with the key negotiated by the algorithm. Nevertheless, in verifying the algorithm, it becomes necessary to violate this separation and perform a check to verify the correctness of the data. To avoid either entity having access to memory locations they should not access, the quantum environment itself checks for the value equivalence and throws an error in the event of their disagreement.

3.1.4 Indeterminable Criteria

The difference between the "failure criteria" and "indeterminable criteria" can be seen by an example: suppose that an eavesdropper attempts to glean information from the quantum connection and causes a failure while calculating Bell's Inequalities. It cannot be known in the experiment's setup if the algorithm was at fault or the eavesdropper was at fault. Using the supposition of a failed reading of Bell's Inequality, we can determine that either a failure in the algorithm or an attempted eavesdropping event occurred. For the purposes of the experiment, this failure actually means that an eavesdropper may have been correctly detected and the protocol was aborted. This should be defined as a success. Nevertheless, the underlying BB84 algorithm has been shown experimentally to detect the eavesdropping entity: the purpose of *this* experiment is to establish identification of the endpoints involved. Since eavesdropper detection has already been shown to be experimentally successful in the existing implementations of BB84, this variable will be controlled. If Bell's Inequalities are not calculated correctly, it either indicates a failure of the entire algorithm or a failure to control the eavesdsropping variable. The former indicates a "failure" scenario while the latter indicates a "success" scenario.

For the purpose of testing and simulation, Bell's Inequality calculations are being provided by simulation, and a failure here will indicate failure of the simulated algorithm.

3.1.5 Expected Error Scenarios

Entropy failures are also expected to occur within the expected frequencies as detailed in Section 2.2.6. Entropy failure scenarios are not a failure of the algorithm; they are a limitation of the quantum environment upon which the algorithm is run. To properly simulate a quantum environment, entropy errors occur when A and B fail to negotiate enough qubits with each other or with T.

Entropy errors that occur outside of the expected range defined in Section 2.2.6 are deemed to be a rejection of the simulation environment upon which the HHUYS16 algorithm is being run. HHUYS16 cannot be accepted in this case. If entropy errors occur at the expected frequency defined in Section 3.3.2, the entropy failure is recorded and the iteration is excluded from meeting the success criteria.

The expected entropy failure rates are further illustrated in Appendix B.

3.2 The Algorithm - HHUYS16

Assumptions: Alice and Bob have registered with Trent and have shared private keys K_A (between Alice and Trent) and K_B (between Bob and Trent) (see Section 3.2.1). Alice (A), Bob (B), and Trent (T) are the actors. Encryption is defined as $\epsilon(x, y)$ where x is the key and y is the message to encrypt. Decryption is defined as $\delta(x, y)$. Padding is defined as $\phi(x, y)$, and unpadding is defined as $\upsilon(x, y)$. It is understood that if any party is unable to unpad or decrypt a message correctly, the protocol is aborted.

1. A sends B a message on the classical channel that she would like to initiate a trusted quantum key distribution.

2. A sends B a series of entangled photons that will become the key.

3. A then announces to T on the classical channel that she would like to communicate securely.

4. T sends A photons on the quantum channel.

5. A and T negotiate the measurement basis as in the BB84 algorithm. The photons sent in step 4 become a temporary pad, P_A, which will be used to pad future messages. P_A will be divided into thirds, P_A^1, P_A^2, and P_A^3.

6. A prepares a message, M_A, for T stating that she would like to communicate with B. This message is first encrypted with K_A and then padded with the first 1/3 of P_A, which is P_A^1. She sends this message to T on the classical channel: $\phi(P_A^1, \epsilon(K_A, M_A))$.

7. T unpads and then decrypts the message
   $(\delta(K_A, \upsilon(P_A^1, \phi(P_A^1, \epsilon(K_A, M_A)))) = M_A)$, then prepares to contact B on behalf of A. T announces on the classical channel to B that he would like to communicate securely.

8. T sends photons on the quantum channel to B.

9. B and T negotiate the measurement basis as in the BB84 algorithm. The photons sent in step 8 become a temporary pad, P_B, which will be used to pad future messages. P_B will be divided into thirds, P_B^1, P_B^2, and P_B^3.

10. B then prepares a message, M_B containing the measurement basis he used to read A's photons in step 2. M_B is first encrypted with K_B, then padded with the first 1/3 of P_B, which is P_B^1. This message is sent to T on the classical channel: $\phi(P_B^1, \epsilon(K_B, M_B))$.

11. T unpads the message containing the measurement basis from B (see step 10) then decrypts it to get M_B using the formula $(\delta(K_B, \upsilon(P_B^1, \phi(P_B^1, \epsilon(K_B, M_B)))) = M_B)$. He then pads it with P_B^3, encrypts it with K_A, then pads it again with P_A^2. This message,
    $\phi(P_A^2, \epsilon(K_A, \phi(P_B^3, M_B)))$, is sent to A on the classical channel. A then unpads and decrypts this, and is left with $\phi(P_B^3, M_B)$ such that
    $\delta(K_A, \upsilon(P_A^2, \phi(P_A^2, \epsilon(K_A, \phi(P_B^3, M_B))))) = \phi(P_B^3, M_B)$.

12. T prepares a message, T_B, with contains the last 1/3 of P_A (which is P_A^3), encrypts it with K_B, and pads it with P_B^2. T sends B the pad P_A^3 over the classical channel as $\phi(P_B^2, \epsilon(K_B, P_A^3))$, which B unpads and decrypts to obtain P_A^3: $\delta(K_B, \upsilon(P_B^2, \phi(P_B^2, \epsilon(K_B, P_A^3)))) = P_A^3$.

13. B pads P_B^3 with the result of step 12, P_A^3, and sends them to A on the classical channel: $\phi(P_A^3, P_B^3)$.

14. A unpads this communication from B to obtain P_B^3: $\upsilon(P_A^3, \phi(P_A^3, P_B^3)) = P_B^3$. A then verifies B's identity through T by performing $\upsilon(P_B^3, \phi(P_B^3, M_B))$ (remember that A has $\phi(P_B^3, M_B)$ from step 11) which will result in the measurement basis M_B. She then completes the negotiation stage of BB84 with B using the measurement basis in M_B.

3.2.1 Registration Phase

The weakest point of this algorithm is the registration phase. This involves a new actor: the network administrator (C). The node being added to the network (A) must inherently trust the administrator (C) adding him to the network.

1. A negotiates the initial private key with T using BB84.

2. T registers A with the negotiated private identification key.

3. A initializes the trusted third-party algorithm with C using the algorithm in Section 3.2.

4. If the algorithm succeeds, A is able to verify that the administrator or node creator, C, has identified a trusted third-party, T.

5. If the algorithm is unsuccessful, the protocol is aborted, and A unregisters from T.

This introduces the possibility of a corrupt network administrator adding nodes to the network who use a malevolent trusted third-party. The nodes that have been added by a good network administrator will maintain their keys negotiated with a good trusted third-party and will be unable to communicate with the bad-faith added nodes.

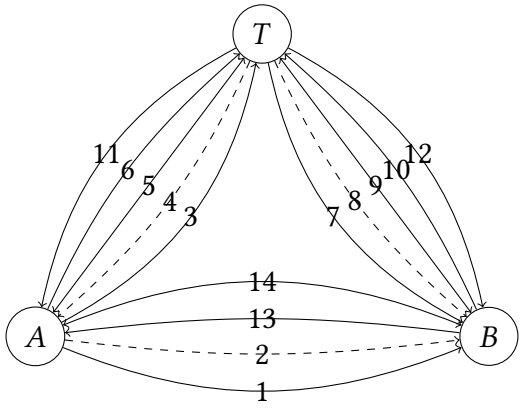

Figure 3.1: Trusted Third-Party Authentication

3.2.2 Negotiation Phase

Figure 3.1 shows the fourteen-step negotiation phase. Note that several of these steps may be performed in parallel (for example, steps 11 and 12). Dashed lines represent data sent on the quantum communication channel; solid lines represent negotiations and communications taking place on the classical communication channel. These fourteen steps correspond to the algorithm described in Section 3.2.

In this negotiation phase, A can only receive the measurement bases from T if T knows A's unique, shared, secret key and has authenticated B. B can only know whom is trying to communicate with him when revealed by T. T must know B's unique, shared, secret key and must have authenticated A in order for B to ever receive this notification.

3.3 Experiment Execution

All functions outside of the sending/receiving of quanta are implemented in package HHUYS16. The package provides an example of the methodology as defined in the algorithm in Section 3.2. The implementation should not be misunderstood as a proof of the algorithm, but rather, as an illustration for showing that the algorithm is complete and can be implemented.

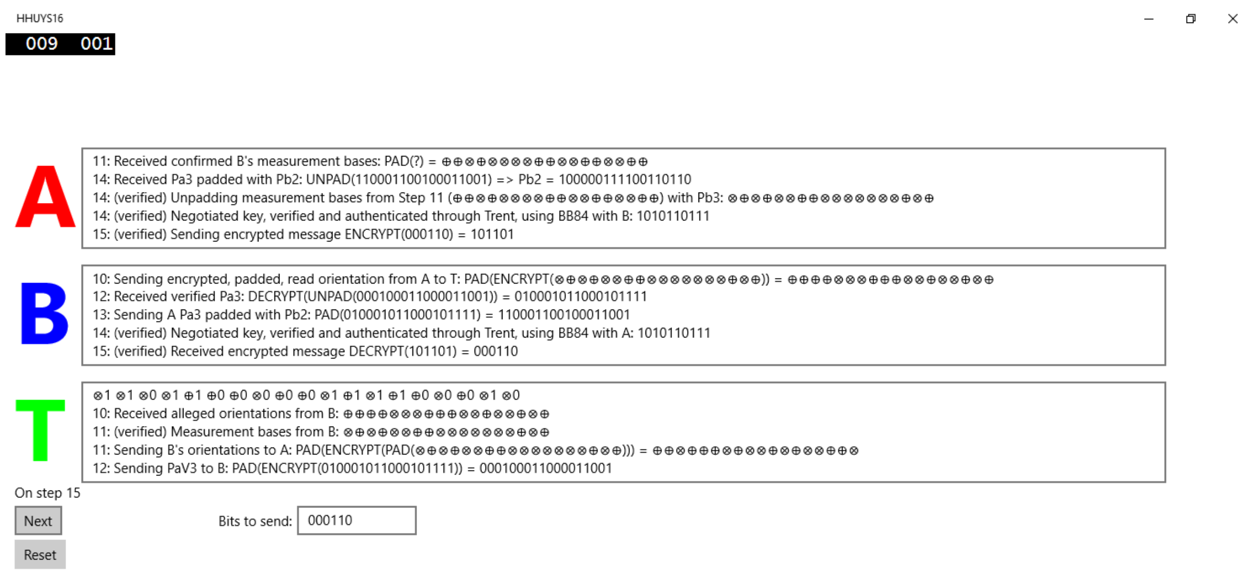

Figure 3.2: Reference Implementation for HHUYS16

Figure 3.2 illustrates and provides an implementation for the steps of the algorithm. The HHUYS16 reference sample implementation is written utilizing the Universal Windows Application 10.0 framework. Depending on only the .NET Core (specifically .NET Core 5) functionality, it can use the .NET open source core libraries to run on any x86, x86_64, or ARM target platforms.

The reference implementation is intended to be tailored to the particular QKD algorithm and negotiation environment of the target deployment and is not intended to be used as-is. Assumptions are made inside the simulated code regarding the BB84 negotiation phase, registration IDs for nodes A and B, and with the encryption algorithm. To simplify the example, encryption and padding are performed bit-at-a-time to allow development of the algorithm in the intended environment.

3.3.1 Simulated Environment and Assumptions

The environment consists of two important pieces: the HHUYS16 Reference Implementation for BB84-based QKD schemes and the simulated BB84 environment to provide quantum entropy in simulating the key exchange.

The implementation uses a simulated environment where a Quanta class provides the simulated BB84 polarizations, intercept capabilities, and negotiation phases. The Quanta class simulates the behavior of entangled quanta and successfully implements the required features of quanta as required by the BB84 and E91 algorithms (Section 2.5.1). Due to the assumption of simulation-controlled measurement of Bell's Inequalities, the measurement flag is sufficient for simulating the entangled quanta's behavior. The padding and encryption algorithms are in their simplest, bit-at-a-time form to afford easier implementation and optimization in an actual quantum environment. It is acknowledged that the bit-at-a-time implementation of these algorithms on a classical processor is not the most efficient solution; however, the simulation attempts to stay consistent with potential quantum implementations.

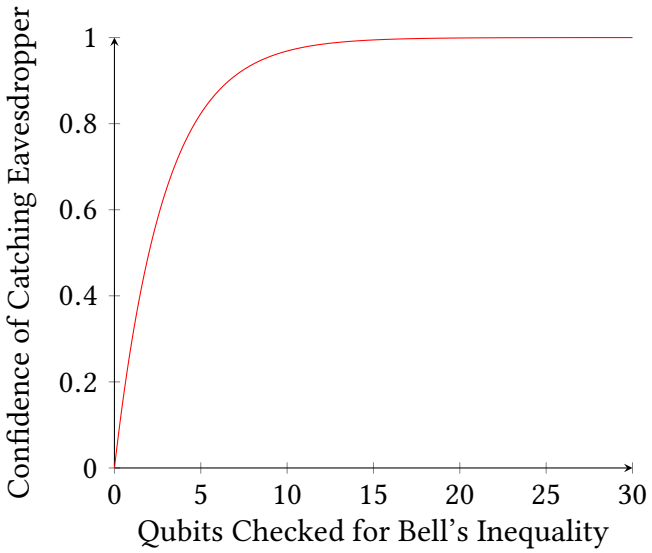

Figure 3.3: Qubits vs Eavesdropper Detection Confidence (per leg)

The largest amounts of quantum data are exchanged in the temporary padding values between T and either endpoint. The validity of the model relies on the ability to catch an eavesdropper between these legs and to keep T from knowing the negotiated key between A and B. Therefore, the pad between T and either endpoint must be greater than or equal to the $3 * n + 26$ where 3 are the number of potential pads and encryptions the key must go through, n represents the number of bits to be encrypted in the original message between A and B, and 26 represents

the number of qubits that must be checked to reach a 99.99% confidence interval in catching an eavesdropper between any leg of the communication. The confidence level of finding an eavesdropper may be adjusted by using Figure 3.3. 99.99% is chosen as the required confidence interval for algorithm acceptance.

Now that a confidence interval of 99.99% has been established in catching an eavesdropper, the largest number of quantum bits that must be exchanged is defined as $3 * n + 26$ where n represents the number of original bits to be negotiated between A and B. Under a binomial distribution, $3 * (3 * n + 26)$ is only able to achieve a 99.99% successful negotiation rate when there are at least 12 bits in the original transmission between A and B. Negotiated padding and keys in the sample implementation are much larger than required; however, when small key sizes are chosen, the potential for the receiving party to read quanta in incorrect bases is much higher. This leaves several options for guaranteeing the algorithm:

- To achieve 99.99% confidence, the minimum payload size between A and B is set at 12 bits.
- A 99.93%—99.987581% confidence is accepted for payload sizes of 1—11 bits.
- To achieve 99.99% confidence, the payload between A and B is padded to at least 12 bits.

The simulation is distributed with minimal default values so that it accepts the lower confidence in the algorithm. This behavior is intended to produce simulation results that encounter error scenarios. Additional simulation runs are included in the results (Section 3.5) that assume a minimum payload size.

A QuantumException class provides notification of a simulated failed reading of Bell's Inequalities. Each endpoint can detect when entropy is insufficient to continue the algorithm; a QuantumException is not triggered in this case, but a failure of the algorithm is recorded. A failure to build up enough entropy results in a reattempted negotiation of the algorithm up to three times.

3.3.2 Model Validation

Using Sargent's criteria for validating simulation models,[68] several methods have been chosen to increase the confidence of the simulation framework. The methods' applicabilities to Balci's verification and validation categorizations are also examined when applicable.[5]

Comparison to Other Models

The Quanta simulation is verified by functional equivalence to the Quantum Model Checker's (QMC) QuantumCoinFlipping routine.[43] The QMC tool is a Java program restricted for use on stabilizer circuits of a universal quantum computation. The proven QMC QuantumCoinFlipping routine[24] is then modified to simulate an entanglement-based quanta by adding a flag to determine if it has been measured.

While semantically equivalent to the QMC QuantumCoinFlipping routine, this simulation is not functionally identical to it. Instead of hardcoding a rounded value for $\sin(45^\circ)$, the simulated environment uses .NET's sine function to calculate a more accurate value for the target platform.

Using what Balci refers to as a static technique,[5] the source code for QMC's QuantumCoinFlipping routine is reverse engineered and analyzed. This syntax analysis verifies that the quantum environment is semantically similar to an already trusted model.

Event Validation

Analyzing the event outcomes and likelihoods can verify that the simulated events intend to reflect the actual quantum environment. By first defining the requirements of the simulated portion of the experiment, a subject-matter expert can assess the correctness of the simulation in compliance with DoDI 5000.61.[27]

The requirements of the simulated portion of the experiment are to generate quantum entropy in a way that is consistent with BB84 and E91's expectation of receiving qubits. The qubit structure and requirements are defined in Section 2.5.1. Since the selection of qubits matches an expected BB84 implementation, and the control of Bell's Inequality measurements matches

the mathematics defined in Section 2.1.2, the discreet events are verified to occur in frequencies consistent with a quantum environment. This piece of verification can also be used as the basis for creating the accreditation documentation required under MIL-STD-3022.[59]

Extreme Condition Tests

Analysis of the algorithm suggests that, on average, it would take $\frac{\ln(50\%)}{\ln((1-(\sin(\pi/4))^n)}$ occurrences of the algorithm before an error is encountered where n represents the number of qubits being negotiated. Appendix B provides a table calculating the average number of runs before a failure occurs. Using Ritter et al's Power Calculation for determining simulation sample size[66], a Cohen coefficient of 0.1, a 99.9% confidence in the algorithm, and 0.1% power\$\alpha$, 3681 runs per parameter variation are required. Note that Excel's NORMINV function is used to calculate the inverse of the cumulative normal distribution of simulated qubits in function Z when Bell's Inequalities are controlled by simulation. Ritter et al suggest using a Cohen coefficient of 0.2 for such edge case testing, but to place the number of runs performed in the category of Sargent's extreme condition testing, a Cohen coefficient of 0.1 is chosen.[21] Balci identifies this as a dynamic validation technique.[5]

$n = 2((Z(1-\alpha/2)+Z(1-1))/d)^2 = 2((Z(1-0.001/2))/0.1)^2 = 2(4.29/0.1)^2 = 3681$

Solving $ln(x)/ln(1-((1/\gamma)^3)) = 3681$ where γ is obtained from the last column of the table in Appendix B, x provides the percent chance of experiencing a failure scenario. The simulation execution should reflect this probability. If the number of entropy failures is much lower or higher than the expected chance of failure, the simulation should be rejected. While it seems counterintuitive to want the simulation to encounter such a failure scenario, understanding that these failures are caused by the quantum environment's failure to build up entropy helps prove the simulation portrays an accurate quantum environment.

Program Walkthrough

A walkthrough of the execution is also provided to further demonstrate the effectiveness of the algorithm but should be considered as an informal illustration of what occurs during each validation execution. Balci identifies walkthroughs as an informal validation method with lower complexity.[5]

Once the Universal Windows application is installed, it will appear as an available application in the Windows Start Menu, titled HHUYS16. Once started, Alice and Bob will register automatically with Trent to set up the requirements of the algorithm. For the sample implementation, a random, binary message is generated for Alice to send to Bob. Resetting the application chooses a new random key for Alice and Bob's registrations with Trent and creates a new message for Alice to send Bob.

A walkthrough of the internal variables and their values is below, numbered in reference to the step which the variable corresponds to in Section 3.2.

0. The message to send: M = 11000010

   K_A = 10110001011000011100100000001001111

   K_B = 11011100010001100000010001110111 0100

2. A sends B (on quantum channel): ⊗1⊗1⊕0⊗0⊗0⊕1⊗0⊗0⊗0⊗0⊗1⊗0⊗1⊗0⊗0⊕1⊗0⊕0

   B receives (on quantum channel): ⊕1⊗1⊕0⊕0⊗0⊗0⊗0⊕0⊕0⊗0⊕0⊕0⊗1⊕0⊗0⊗1⊕0⊕0

4. T sends ⊕1⊗1⊗0⊗1⊗0⊗0⊗0⊗0⊗1⊗0⊗0⊕1⊗1⊕1⊗1⊕1⊗1⊕0⊗1⊕1⊗1⊕0⊗0⊕0⊗0⊗
   0⊗0⊗1⊗1⊗0⊕0⊗0⊕1⊕0⊗0⊕0⊗1⊗0⊕0⊕1⊕0⊕1⊗1⊗0⊕0⊗0⊗1⊕0⊗0⊗1⊕0⊕0⊕1⊗
   0⊗0⊕1⊗0⊕0⊕1⊗1⊕0⊗0⊗1⊗1⊕0⊗1⊗0⊗1⊗1⊕1⊕1⊕0⊗1⊕1⊕0⊕1⊗1⊗1⊗0⊕0⊕0⊗
   0⊗1⊕1⊕1⊕0⊗0⊗0⊗1⊗1⊗0⊗1⊗1⊕0⊕1⊕0⊕1⊕0⊕1⊗1⊗0⊕0⊗1⊗1⊗1⊕0⊗0⊕0⊗1
   A receives ⊕1⊕1⊕0⊗1⊗0⊗0⊗0⊗1⊕0⊗0⊗1⊗1⊗1⊕1⊗0⊗1⊗0⊕1⊕1⊗1⊕0⊕0⊗0⊕1⊕
   0⊗0⊕1⊗1⊗0⊕0⊕1⊗0⊗1⊕1⊕0⊕0⊕0⊗1⊗1⊗0⊕1⊗1⊕1⊕0⊕1⊕1⊕0⊕0⊕1⊗0⊕0⊕1⊗
   0⊕0⊗1⊕0⊕0⊕1⊗1⊗0⊕1⊕0⊕0⊕0⊕0⊕0⊕1⊗1⊕1⊗1⊕0⊗1⊗0⊕0⊗1⊕0⊗1⊕0⊕0⊕0⊕
   0⊗1⊗0⊗1⊕0⊕0⊕1⊗1⊗1⊗0⊗1⊗1⊕0⊗0⊕0⊗1⊕0⊕1⊗1⊕1⊗1⊕0⊕1⊗1⊗0⊗0⊕0⊗1

5. Negotiated P_A = ⊕1⊗1⊗0⊗0⊗0⊗1⊗0⊗1⊗1⊕1⊗1⊕0⊗0⊗1⊗0⊕0⊕0⊕1⊗1⊕0⊕0⊕0⊕1⊗
0⊕0⊕1⊗1⊕0⊗1⊕1⊕0⊗1⊕0⊗1⊕0⊕0⊗1⊕0⊗1⊗1⊗0⊗1⊗1⊕0⊕0⊕0⊕1⊗1⊗1⊗0⊕0⊗1

6. *A* sends *T* the comm packet "01" which is encrypted to "00" then verified using decryption by *T*.

8. *T* sends *B*: ⊗0⊗0⊕0⊗0⊗0⊕1⊕1⊕0⊗0⊕1⊗1⊗1⊗0⊗0⊕0⊕1⊕0⊗0⊕0⊗1⊕0⊗0⊕1⊕1⊗
1⊗1⊗1⊕1⊕0⊕0⊕0⊕1⊗1⊗1⊗1⊕1⊗1⊗0⊗1⊗1⊕0⊕0⊗0⊕0⊗1⊗1⊕0⊕1⊕0⊕0⊗1⊗1⊗
1⊗0⊗1⊗1⊗0⊕0⊗0⊗0⊗0⊗1⊕1⊗0⊕0⊕0⊗1⊗1⊕1⊕0⊗1⊕0⊗0⊗1⊗0⊕0⊕1⊗1⊕0⊗1⊕
1⊕1⊗0⊕1⊗1⊕0⊕0⊕0⊗1⊕0⊗0⊕0⊕0⊕0⊕0⊕0⊗0⊗1⊗1⊕1⊗1⊗1⊗0⊕1⊗0⊕1⊗1⊕1
B receives: ⊕1⊕0⊕0⊕0⊕0⊕1⊕1⊗0⊕1⊕1⊗1⊗1⊕0⊕0⊗0⊕1⊗0⊕0⊕0⊗1⊗0⊗0⊗1⊕1⊗
1⊕1⊗1⊕1⊕0⊕0⊕0⊕1⊗1⊕1⊕1⊗0⊗1⊕0⊕1⊗1⊗1⊕0⊗0⊗1⊕0⊕1⊗0⊗0⊗1⊗0⊕1⊕1⊗
1⊕0⊕0⊗1⊕0⊕0⊗0⊕0⊕0⊕1⊗1⊗0⊕0⊕0⊗1⊕1⊗1⊕0⊗1⊗0⊕1⊗1⊕1⊕0⊕1⊕1⊕0⊗1⊗
0⊕1⊗0⊗0⊗1⊗1⊗0⊗1⊗1⊗0⊗0⊗0⊗0⊗0⊗0⊕0⊗0⊕1⊕1⊗1⊗1⊗1⊗0⊗1⊗0⊕1⊕0⊗0

9. Negotiated P_B = ⊕0⊕1⊕1⊕1⊗1⊗1⊕1⊕0⊗1⊗0⊕1⊗1⊗1⊕1⊕0⊕0⊕0⊕1⊗1⊗1⊗1⊕0⊗
0⊗1⊗1⊕0⊗0⊗0⊕0⊕0⊗1⊕0⊗1⊗1⊕0⊕1⊕0⊗1⊕1⊗0⊗1⊗1⊗0⊕0⊗0⊗1⊗1⊗0⊗0⊕1

10. *B* encrypts and pads the orientations to *T* $PAD(ENCRYPT($⊕ ⊗ ⊕ ⊕ ⊗ ⊗ ⊗ ⊕ ⊕ ⊗ ⊕ ⊕ ⊗ ⊕ ⊗ ⊗ ⊕⊕)) = ⊗ ⊗ ⊗ ⊕ ⊗ ⊗ ⊕ ⊕ ⊗ ⊕ ⊗ ⊗ ⊕ ⊕ ⊕ ⊗ ⊕⊗

11. *T* pads, encrypts, and pads the orientations to *A* $PAD(ENCRYPT(PAD($⊕ ⊗ ⊕ ⊕ ⊗ ⊗ ⊗ ⊕ ⊕ ⊗ ⊕ ⊕ ⊗ ⊕ ⊗ ⊗ ⊕⊕))) = ⊕ ⊗ ⊗ ⊕ ⊕ ⊕ ⊗ ⊗ ⊗ ⊕ ⊕ ⊗ ⊕ ⊗ ⊗ ⊗ ⊗⊗

12. *T* sends P_A^3 to *B*: $PAD(ENCRYPT(100101101100011100))$ = 001000011110001000
B receives P_A^3 = $DECRYPT(UNPAD(001000011110001000))$ = 100101101100011100

13. *B* sends *A* $PAD(100101101100011100)$ = 011100001100110001

14. *A* verifies $UNPAD(011100001100110001)$ => $Pb2$ = 111001100000101101
A uses the verification to unpad Step 11: $UNPAD($⊗ ⊕ ⊗ ⊕ ⊗ ⊕ ⊕ ⊕ ⊕ ⊗ ⊕ ⊕ ⊕ ⊕ ⊕ ⊕ ⊕⊗) =

$\oplus \otimes \oplus \oplus \otimes \otimes \otimes \oplus \oplus \otimes \oplus \oplus \otimes \oplus \otimes \otimes \oplus\oplus$

A and B negotiate the key 10000100

15. A sends B the encrypted message:

    $ENCRYPT(11000010) = 01000110$

    $DECRYPT(01000110) = 11000010$

This sample runthrough is taken directly from the HHUYS16 reference example interface program. The program provides more verbosity through each stage of the process to help a developer implement the methodology here.

3.4 Complexity and Cryptanalysis

Complexity for the algorithm is heavily dependent upon the implementation of the underlying QKD algorithm and underlying encryption algorithm of HHUYS16. Assuming the use of BB84 and the Vernam One-Time Pad for these algorithms, the basis for their complexity can be used to calculate the complexity of HHUYS16. Furthermore, a cryptanalysis of HHUYS16 is provided to examine the security improvements.

3.4.1 Complexity

Assuming that an iteration of the BB84 algorithm is a single calculation of complexity, the new trusted third-party algorithm has the following complexities:

- A single BB84 instance between A and B: n.
- A double-sized BB84 instance between A and T: $2n$.
- A double-sized BB84 instance between B and T: $2n$.
- The overhead of the packet that identifies the sender and recipient between A and T and B and T: $2c$. Note that the size of the packet may be constrained by limiting the size of registration names in the network. If IPv6 names are used, this packet could be constrained

to a size of 256 bits of extra needed padding key length. Since both the legs between $A \rightarrow T$ and $T \rightarrow B$ communication channels must encrypt separate packets of this size, the complexity assumes that 512 bits of padding for these packets is sufficient.

Therefore, the communication complexity of this algorithm is $O(5n)$ where n represents the number of bits needed in the negotiated cryptographic key using the BB84 algorithm, and an additional constant complexity of 512 bits of padding is needed if the network is of type IPv6 and the IPv6 addresses are used as the quantum identifiers.

A more meaningful complexity analysis comes by incorporating the additional complexity into the Kolmogorov complexity calculations of BB84. Starting with Miyadera and Imai's formula, the additional overhead can be added.[60]

- Let the Kolmogorov complexity of BB84 be defined as $M \simeq N(1 - h(2(p + \epsilon)))$ where h is the Shannon entropy calculation, p is the pre-agreed upon error threshold, and ϵ is an additional security parameter used to increase the eavesdropper detection rate of BB84 as defined by Miyadera and Imai.
- Let t represent the maximum number of bits to identify a node in the network.
- The Kolmogorov complexity of the trusted, third-party authenticated algorithm is defined as $M \simeq 5N(1 - h(2(p + \epsilon + t)))$.

3.4.2 Cryptanalysis

The algorithm follows the same cryptanalysis as the base BB84 algorithm, relying on its initial base calculation to determine at which leg E will attempt to compromise the quantum negotiation stages.[54] Further calculations and an alternative to the BB84 algorithm are available in Section 2.5.1.

Proof. Exhaustive.

- Suppose that Alice, A, is attempting to establish a quantum key with Bob, B, around an eavesdropper, Eve, E.
- E attempts to intercept a single photon and must choose an orientation to read it in, $\oplus$ or $\otimes$. E has a 50% chance of selecting the correct orientation.
- If E chooses the correct orientation, she has a 100% chance of intercepting the sent photon.
- If E chooses the incorrect orientation, she has a $\sin(45^\circ)$ chance of intercepting the correct value, and B has a $\sin(45^\circ)$ chance of reading the resent photon in the wrong orientation from E, resulting in a $\sin^2(45^\circ)$ (or 50%) chance of getting caught.
- E has a 75% chance of intercepting a single photon without detection (exhaustive proof of all four equally possible outcomes).
- The chance of getting caught is $1 - (\frac{3}{4})^n$ where n are the number of bits transferred on the quantum channel.

□

It is important to remember that the chances of being caught at each leg are not mutually exclusive. Being caught between the A and T nodes voids the algorithm in the same way that getting caught between A and B does. Therefore, E cannot improve her chances by eavesdropping upon multiple legs of quanta. This is an important distinction to make between classical cryptanalysis and quantum cryptanalysis: E does not improve her chances of intercepting the negotiated key by listening in to multiple legs of the negotiation; she increases her chance of getting caught. Therefore, it is better for her to choose the single leg most likely to accomplish her goal.

In order for E to compromise the key without detection in the trusted third-party protocol, she must accomplish one of the following:

1. read the measurement bases sent between B and T or T and A and intercept the initial quantum negotiation between A and B

2. compromise C and impersonate T when A and B are placed on the network and intercept the quantum negotiation between A and B

3. compromise the shared secret keys for A and B, impersonate T, and intercept the quantum negotiation between A and B

Compromising Measurement Bases

E can compromise the secret key between either B and T or T and A to determine the measurement bases of the key negotiated between A and B. The measurement bases are first encrypted with the shared secret registration key, then they are padded with the negotiated session key between either A or B and the trusted, third-party T. To obtain the measurement bases, E must first compromise the shared secret key of one of the nodes. This is done by compromising the registration phase (see Section 3.4.2) or by intercepting the key during registration. Since the registration phase relies on a single iteration of the QKD algorithm, the chance of successfully compromising the shared secret key is $1 - (\frac{3}{4})^n$ where n is the size of the shared private key.

At this point, E has the necessary requirements for impersonating a node in the network and violating the integrity of the node; however, if the goal is to intercept the communication without getting caught, E must also compromise the session key between that node and T. Assuming the session key size is m, E can compromise this key with a probability of $(\frac{3}{4})^m$.

Compromising both keys therefore results in a chance of intercepting the key without detection of $(\frac{3}{4})^{n+m}$.

Compromising the Registration Stage

Assuming that E acts as a pernicious C in adding A and B to the network, she can define herself as T and violate the integrity of the trusted third-party algorithm with no major effort. At this point, the communication between A and B enters the realm of the standard BB84 protocol; undetected compromise by both parties occurs with a likelihood of $(\frac{3}{4})^n$ where n is the size of

the distributed key. E gains the ability to impersonate any node that uses her pernicious T actor (which could also be herself) as its trusted third-party.

To compromise a new node's registration key, E must detect it during its negotiation with T. This occurs with a probability of $(\frac{3}{4})^m$ where m is the shared private key's size.

E's easiest path to compromise communication between A and B undetected requires that she impersonate both C and T while adding A and B to the network. She can then impersonate either A or B in communication between the two. To detect communication between the two without impersonation or detection, she must detect the initial quanta and the chance of her success falls to the underlying QKD algorithm (in BB84's case, $(\frac{3}{4})^n$ where n are the number of bits in the key).

Impersonating the Trusted, Third-Party

Perhaps the most difficult possibility of compromise comes in attempting to impersonate T without acting as a pernicious C. This requires both secret keys of A and B to be compromised, shown to be $(\frac{3}{4})^m$ in Section 3.4.2. To compromise both keys, the probability of success for E is $(\frac{3}{4})^{n^2}$. While it is easier for E to impersonate one of the nodes at this point, we assume that E needs to compromise the communication between A and B without detection.

E must then compromise the original quantum negotiation between A and B (shown previously to be a probability of $(\frac{3}{4})^m$). The chance of success for accomplishing this exploit is then determined to be $(\frac{3}{4})^{n^2+m}$.

3.5 Experiment Results

Recall from Section 3.3.1 that the binomial distribution for very small key payload sizes in the algorithm is less than 99.99%. A 99.99% confidence interval is only available for negotiated key sizes of 12 bits or more. When negotiating a 100-bit key, this confidence interval increases to 99.9999999987%.

By design, there are failure scenarios in the algorithm. A failure in the reading of Bell's Inequalities will notify the communicating parties of compromise. A failure of building up enough entropy will alert the communicating parties of a hardware fault, an attempted compromise of the communication, or that the parties did not agree on enough measurements.

The entropy of the algorithm is part of the simulation: it's a basic requirement of quantum coin flipping for building the quantum key. A failure of the entropy, however, does not reflect on a failure of the algorithm; it is a reflection of the quantum environment due to the uncertainty imposed in picking measurement bases. The simulation will therefore capture the failures caused by quantum uncertainty. Failures to build up enough entropy should occur within the expected confidence interval found in Section 3.3.1. Repeated failures outside of the selected confidence interval can be assumed as either hardware failure or malicious intent. The algorithm suggests up to three retries when an entropy failure is encountered. It is equally likely to bruteforce the key on the first guess than it is to fail the algorithm three times under normal conditions. A failure three times in a row suggests hardware failure with a potential of malicious intent and should be investigated.

The experiment was executed 3681 times with each number of qubits of entropy from 27 to 100. One entropy failure was recorded at value 28 (26 bits of Bell's inequalities and 2 bits of key size), and one entropy failure was recorded at value 30 (26 bits of Bell's inequalities and 4 bits of key size). When implementing the three-retries policy, the experiment was rerun, and no entropy errors were encountered. To calculate the average number of runs required before encountering an error when implementing the three retries policy, the formula $ln(0.5)/ln(1 - ((1/\gamma)^3))$ can be used where γ is obtained from the last column of the table in Appendix B. The failure rate is therefore consistent with the expected failure rates described in Section 3.3.2.

Primarily, the experiment shows that A and B can establish a trust relationship with each other based on their mutual trust of T. Integrity is therefore added to the existing confidentiality guarantees; HHUYS16 establishes a high confidence in the confidentiality and integrity of the negotiated key.

To achieve high degrees of certainty in the success of the algorithm (IE: affecting its availability), the author recommends negotiating key sizes of at least 100 bits while checking Bell's Inequality on 29 bits. Using these parameters, the simulation was performed over one million times using random distributions, and no failure scenarios and no retry attempts were encountered. The chances of failure in this scenario are well below any meaningful accepted limits. Increases to the checking of Bell's Inequalities will yield a higher confidence in detecting an eavesdropper, but an increase in the distribution of qubits checked for Bell's Inequalities yields a lower distribution of qubits dedicated to negotiating the key or a requirement to exchange more qubits on the quantum channel.

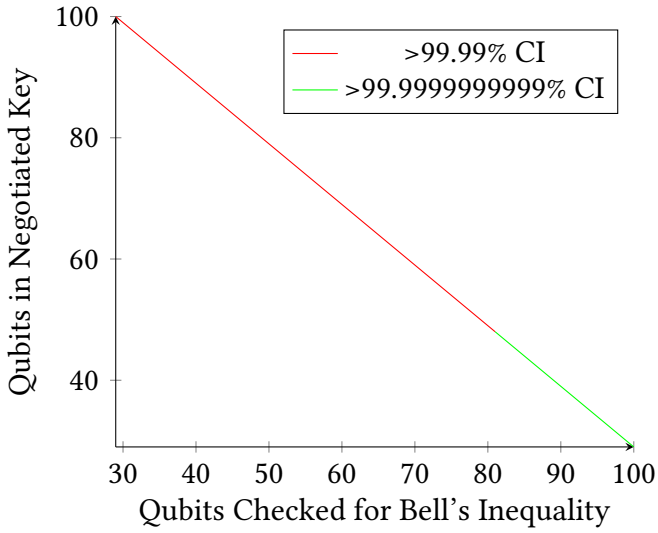

Figure 3.4: Qubit Distribution for 99.9999999999% CI of Algorithm Success

Figure 3.4 shows the distribution of negotiated qubits associated with measuring for Bell's Inequalities and associated with negotiating the quantum key. The red line indicates a >99.99% probability of detecting an eavesdropper, and the green line indicates a >99.9999999999% probability of detecting an eavesdropper. The entire line represents the lower boundary for achieving a 99.9999999999% confidence interval in the algorithm's successful completion and full acceptance of the algorithm and sample implementation.

A graphical implementation of the simulation and algorithm is available at `https://www.hoodsecurity.com/Jon/` to help others reproduce the results that are obtained here. Dependencies and a walkthrough of the simulation are available in Section 3.3.2.

Chapter 4

Future Work

The algorithm in its current form assumes the use of BB84 and Vernam One-Time Pads. Other algorithms may be used once they have been evaluated for quantum resistance. Methods for extending HHUYS16 based off of preliminary research into these quantum-resistant algorithms are included here, though lack of proofs for the proposed new algorithms should prevent them from being implemented in a production environment. The new algorithms are believed to be quantum-resistant, but no proofs exist to verify these claims. Legwork for authorizing such systems on DoD networks has been performed in this dissertation, but future work remains to approve the guidelines and documentation at a governmental level.

4.1 Extending the Algorithm

The algorithm may be modified to optimize its payloads by implementing algorithms other than one-time pads.

4.1.1 Using NTRUSign

The encryption function of the algorithm verifies the shared, private key negotiated between each endpoint and the trusted third party. Simply replacing the encryption function with NTRUSign is insufficient. After several signatures, enough data is leaked about the private key to render it compromised. Instead, Trent and each endpoint can negotiate an optimal lattice, and the encryption-padding stage can be replaced by negotiating a vector to calculate relationships within the lattice. Instead of needing 1/3 of the QKD bits to pad the encrypted communication, all that is needed are enough bits to define a random vector in relationship to the lattice and enough

bits to pad the solution while in transit (avoiding data leakage about the prenegotiated lattice). The HHUYS16 algorithm becomes the following:

Assumptions: Alice and Bob have registered with Trent and have shared private lattices L_A (between Alice and Trent) and L_B (between Bob and Trent), and they have registered a particular vector within the lattice, V_T^A (between Alice and Trent), and V_T^B (between Bob and Trent) (see Section 3.2.1). Alice (A), Bob (B), and Trent (T) are the actors. Padding is defined as $\phi(x, y)$, and unpadding is defined as $\upsilon(x, y)$. Hash-signing is defined as $\sigma(L, V, T)$ where L is the lattice, V is the vector to measure against the lattice, and T is the contents to hash+sign. Verifying the hash+sign is defined as $\chi(L, V, S, M)$ where L is the lattice, V is the vector, and S is the signature to verify, and M is the message associated with the signature. A vector creation function, $\lambda(R)$, can take a random number, R, and turn it into a vector within the lattice space. It is understood that if any party is unable to unpad, decrypt, or verify a signed message correctly, the protocol is aborted.

1. A sends B a message on the classical channel that she would like to initiate a trusted quantum key distribution.

2. A sends B a series of entangled photons that will become the key.

3. A then announces to T on the classical channel that she would like to communicate securely.

4. T sends A photons on the quantum channel.

5. A and T negotiate the measurement basis as in the BB84 algorithm. The photons sent in step 4 become a temporary pad and random vector, P_A, which will be used to pad future messages and define the vector for future lattice functions. P_A is divided into three pieces, P_A^0, P_A^1, and P_A^2.

6. A sends a message, M_A, to T that she would like to communicate with B. This message is concatenated with the digital signature and then padded to send it across the classical channel: $\phi(M_A + \sigma(L_A, V_T^A, M_A), P_A^0)$

7. T unpads the message from step 6, obtaining $\upsilon(\phi(M_A + \sigma(L_A, V_T^A, M_A), P_A^0), P_A^0) = M_A + \sigma(L_A, V_T^A, M_A)$. T verifies A's signature using verification function $\chi(L_A, V_T^A, \sigma(L_A, V_T^A, M_A))$. T announces to B on the classical channel that he would like to communicate securely.

8. T sends B photons on the quantum channel.

9. B and T negotiate the measurement basis as in the BB84 algorithm. The photons sent in step 8 become a temporary pad and random vector, P_B, which will be used to pad future messages and define the vector for future lattice functions. P_B is divided into three pieces, P_B^0, P_B^1, and P_B^2.

10. B sends T a message, M_B, that contains his measurement basis from step 2. This message is then concatenated with the message's signature then padded with P_B^0 so that T receives $\phi(M_B + \sigma(L_B, V_T^B, M_B), P_B^0)$ on the classical channel.

11. T obtains the original message, M_B by performing $\upsilon(\phi(M_B + \sigma(L_B, V_T^B, M_B), P_B^0), P_B^0) = M_B + \sigma(L_B, V_T^B, M_B)$ with the message from step 10. T verifies the digital signature by performing $\chi(L_A, V_T^B, \sigma(L_B, V_T^B, M_B))$. Now that T has verified B's identity, he prepares a message for A containing the measurement basis from B padded with P_B^1, prepended to the digital signature of the padded message, then padded again with P_A^1: $\phi(\phi(M_B, P_B^1) + \sigma(L_A, V_T^A, \phi(M_B, P_B^1)), P_A^1)$.

12. T sends B part of A's temporary padding data on the classical channel and appends the signature of the message to it: $\phi(P_A^2 + \sigma(L_B, V_T^B, P_A^2), P_B^2)$

13. A then obtains $\phi(M_B, P_B^1)$ from step 11 and verifies the signature from T (now T and A are both validated with each other, and B has validated himself with T).

14. B obtains P_A^2 and verifies the signature from step 12 (B has now validated T, meaning all but the connection between A and B is validated). B then prepares a classical-channel message for A: $\phi(P_B^1, P_A^1)$

15. A obtains P_B^1 from step 14. A uses P_B^1 to obtain the measurement bases, M_B, from step 13. B knows that if A gets the correct measurement bases, they were sent because T, whom B trusts, sent them to A, whom he trusts. A knows that if B was able to pad the measurement bases with P_A^1, he was sent that information by T who verified his trust with B. A and B complete the BB84 negotiation phase on the classical channel using the measurement bases, M_B.

The padding of the vector and signing are critical here: each digital signature with NTRUSign leaks some data about the underlying lattice. By padding the relationships to this lattice, the longevity of it is increased. No data are leaked by an intercepting party. The only way someone can get information about the lattice is by impersonating Trent and completing the algorithm up to the signing stages. The first time the algorithm fails at this stage, the endpoints attempting to authenticate through Trent will know that there was a failure of the algorithm. It will take several thousand failures before enough information is leaked to reconstruct portions of the lattice.[36] Since the data are padded, it is unlikely for this to occur, and the high number of required failures will provide sufficient space for random failures and to alert the nodes that Trent has been compromised.

4.1.2 Extending the Life of the Lattice

The nature of NTRUSign means that it leaks data about the underlying lattice with each signature. The algorithm defined in Section 4.1.1 is sufficient for verification purposes, but an attacker can quickly exhaust the information about the lattice since the signing phase from T to B starts with the same initialization vector, V_T^B. The algorithm can randomize this initialization vector by padding it with additional negotiated random data. It does not stop an attacker from trying to figure out the lattice, but it does extend the length of the lattice sufficiently to know that one of the endpoints has been compromised. Using the same assumptions as Section 4.1.1:

1. A sends B a message on the classical channel that she would like to initiate a trusted quantum key distribution.

2. A sends B a series of entangled photons that will become the key.

3. A then announces to T on the classical channel that she would like to communicate securely.

4. T sends A photons on the quantum channel.

5. A and T negotiate the measurement basis as in the BB84 algorithm. The photons sent in step 4 become a temporary pad and random vector, P_A, which will be used to pad future messages and define the vector for future lattice functions. P_A is divided into four pieces, P_A^0, P_A^1, P_A^2, and P_A^3.

6. A sends a message, M_A, to T that she would like to communicate with B. This message is concatenated with the digital signature and then padded to send it across the classical channel: $\phi(M_A + \sigma(L_A, \lambda(\phi(V_T^A, P_A^3)), M_A), P_A^0)$

7. T unpads the message from step 6, obtaining $\upsilon(\phi(M_A + \sigma(L_A, \lambda(\phi(V_T^A, P_A^3)), M_A), P_A^0), P_A^0) = M_A + \sigma(L_A, \lambda(\phi(V_T^A, P_A^3)), M_A)$. T verifies A's signature using verification function $\chi(L_A, \lambda(\phi(V_T^A, P_A^3)), \sigma(L_A, \lambda(\phi(V_T^A, P_A^3)), M_A))$. T announces to B on the classical channel that he would like to communicate securely.

8. T sends B photons on the quantum channel.

9. B and T negotiate the measurement basis as in the BB84 algorithm. The photons sent in step 8 become a temporary pad and random vector, P_B, which will be used to pad future messages and define the vector for future lattice functions. P_B is divided into four pieces, P_B^0, P_B^1, P_B^2, and P_B^3.

10. B sends T a message, M_B, that contains his measurement basis from step 2. This message is then concatenated with the message's signature then padded with P_B^0 so that T receives $\phi(M_B + \sigma(L_B, \lambda(\phi(V_T^B, P_B^3)), M_B), P_B^0)$ on the classical channel.

11. T obtains the original message, M_B by performing $\upsilon(\phi(M_B+\sigma(L_B, \lambda(\phi(V_T^B, P_B^3)), M_B), P_B^0), P_B^0) = M_B + \sigma(L_B, \lambda(\phi(V_T^B, P_B^3)), M_B)$ with the message from step 10. T verifies the digital signature by performing $\chi(L_A, \lambda(\phi(V_T^B, P_B^3)), \sigma(L_B, \lambda(\phi(V_T^B, P_B^3)), M_B))$. Now that T has verified B's identity, he prepares a message for A containing the measurement basis from B padded with P_B^1, prepended to the digital signature of the padded message, then padded again with P_A^1: $\phi(\phi(M_B, P_B^1) + \sigma(L_A, \lambda(\phi(V_T^A, P_A^3)), \phi(M_B, P_B^1)), P_A^1)$.

12. T sends B part of A's temporary padding data on the classical channel and appends the signature of the message to it: $\phi(P_A^2 + \sigma(L_B, \lambda(\phi(V_T^B, P_B^3)), P_A^2), P_B^2)$

13. A then obtains $\phi(M_B, P_B^1)$ from step 11 and verifies the signature from T (now T and A are both validated with each other, and B has validated himself with T).

14. B obtains P_A^2 and verifies the signature from step 12 (B has now validated T, meaning all but the connection between A and B is validated). B then prepares a classical-channel message for A: $\phi(P_B^1, P_A^1)$

15. A obtains P_B^1 from step 14. A uses P_B^1 to obtain the measurement bases, M_B, from step 13. B knows that if A gets the correct measurement bases, they were sent because T, whom B trusts, sent them to A, whom he trusts. A knows that if B was able to pad the measurement bases with P_A^1, he was sent that information by T who verified his trust with B. A and B complete the BB84 negotiation phase on the classical channel using the measurement bases, M_B.

In this case, every signature vector is padded in a way that prevents the same initialization vector from being used for each session of the algorithm. This increases the lifespan of the underlying lattice.

4.1.3 An Updated Registration Phase

The registration phase of the algorithm can be updated to support a random, shared lattice. As in Section 3.2.1, the network administrator (C) should be involved. The following steps attempt to register A with the trusted quantum network:

1. A negotiates the initial lattice edges, L_A, and a random vector, V_T^B, using BB84.
2. T registers A with the negotiated private lattice and vector.
3. A initializes the trusted third-party algorithm with C using the algorithm in Section 4.1.1.
4. If the algorithm succeeds, A is able to verify that the administrator or node creator, C, has identified a trusted third-party, T.
5. If the algorithm is unsuccessful, the protocol is aborted, and A unregisters from T.

4.2 Implementation Guidance and Recommendations

System management personnel often do not keep up with the latest research on security. Often, an IA analyst or manager will fall into the trap of "checkbox security." In this mindset, the IA departments of an organization obtain a set of minimum requirements, then check off each of the requirements as they are implemented. This attitude towards security has several flaws, but has been inadvertently encouraged by the United States Government (USG), particularly in DIACAP (see Section 1.1).

Under DIACAP, USG systems had to proceed through a validation process that included securing assets in accordance with Security Technical Implementation Guidelines (STIGs). System owners would download the STIG checklists every 3 years, make sure they follow most of them, then completely ignore the security of their systems until time for reaccreditation. In contrast to DIACAP, RMF requires a certain degree of continuous monitoring to avoid the problem of "checkbox security."[73, p. 2, 31]

The Defense Information Systems Agency (DISA) publishes the official set of STIGs on the IASE website: `http://iase.disa.mil/`. Systems are still required to abide by the minimum standards published in the STIGs.

Three important pieces are missing in the application of QKD systems within RMF:

1. guidance on Quantum-Resistant Encryption Algorithms
2. proper categorization of QKD systems
3. a QKD STIG

This section attempts to address each of these items.

4.2.1 Quantum-Resistant Encryption Algorithms

When implementing cryptography, systems are required to use cryptographic standards authorized by NIST. NIST maintains the list of FIPS-140-2 authorized encryption algorithms[41] at `http://csrc.nist.gov/groups/STM/cavp/validation.html`. Currently, this list does not identify the quantum-resistant algorithms. In June 2015, NIST agreed to work on guidance in a NIST Interagency Report (NISTIR) to address this particular issue.

The biggest problem is that NIST is trying to establish a new set of standards when dealing with quantum-resistant encryption algorithms. Instead, the existing system should be used. Algorithms that are not quantum-resistant should be removed from authorization, and only quantum-resistant algorithms should remain. The reason why this is not done is because it would break existing encryption schemes, including AES and TLS. Nevertheless, NIST should put a schedule out for when the government is required to switch to quantum-resistant algorithms. Several algorithms, such as Microsoft's R-LWE[13], have been shown to be quantum-resistant. This standard is even loadable as a module in the current TLS versions.

Short-Term Solution

As a short-term solution, NIST should immediately add a field to the list of FIPS-140-2 authorized algorithms indicating which ones are quantum-resistant. A timeframe of when the non-resistant algorithms will be expired should be established to give everyone time to move to the new algorithms. Alternatively, a new FIPS standard should be created with the goal of phasing out FIPS-140-2. NIST should require that any national security system categorized under RMF as High for either the Confidentiality or Integrity attributes implement the new encryption standards immediately.

FIPS-171, the federal standard for key distribution mechanisms, should be updated immediately to allow for Quantum Key Distribution as a key distribution scheme.

PKI authentication (as implemented with PIV and CAC) class 3 and higher should support both the existing PKI infrastructure and an updated version of the infrastructure implementing a quantum-resistant algorithm.

Moderate-Term Solution

NIST guidance 800-57 Revision 3 (2012) details key management techniques and references FIPS and NIST authorized algorithms for each stage of the cryptographic process.[6] As a moderate-term solution, NIST Publication 800-57 should be revised to address quantum cryptography. Section 3.1 should add QKD schemes such as BB84 as a permitted method for safeguarding confidentiality of data. Section 3.2 should address the checking of Bell inequalities as a permitted method of addressing integrity. As a subset of integrity, authentication and authorization sections 3.3 and 3.4 should permit schemes such as this thesis' trusted third party authentication scheme (Section 3.2) as a permitted method.

Such a moderate-term solution would align FIPS recommendations from the short-term solution with NIST guidance. Since existing STIG and RMF requirements for the DoD reference NIST Publication 800-57 and FIPS-140, updating these guidances would have an immediate impact on DoD-wide systems. If recommendations and guidance expire non-quantum-resistant algorithms,

legacy systems (and nearly every cryptographic system DoD-wide) would need to be updated to support the new recommendations. To alleviate some of the burden on the system owners and developers, the long-term solution should be implemented.

CNSSI guidance should be double-checked to verify that references to the new versions of FIPS-140 and NIST 800-57 do not cause a conflict, and QKD solutions that are properly vetted to implement the recommendations should be NSA-authorized and classified as a Type 1 encryption device.

Long-Term Solution

As a long-term solution, NIST should expire the non-resistant algorithms authorized under FIPS-140-2, and a new competition for the "next AES" should be created. A scaled timeline, depending on a system's RMF categorization level, should be established for when the phase over to the new algorithms should be completed.

All PKI implementations should no longer support existing RSA and AES keys, instead only supporting the quantum-resistant keys implemented in the short-term solution.

4.2.2 Proper Categorization of QKD Systems

QKD systems are not currently permitted by any DoD guidance (they are also not explicitly disallowed). Permitting QKD for use in DoD systems would enable adoption of more secure cryptography. In addition to permission, a recommendation that QKD is used for systems with a High RMF categorization of Confidentiality should be made.

A revision to CNSSI 4009 should be made to permit QKD in certain types of encryption devices. Please note that specific recommendations at this level should be in classified environments.

While permitting QKD in national security systems is relatively easy, securing the methods the QKD implementation uses can be very difficult. As such, guidance should be published as to the minimum QKD implementation standards.

4.2.3 Security Technical Implementation Guide for QKD

Without published guidance, QKD may be implemented in an insecure or unsafe manner. As such, a STIG should be created detailing the minimum standards to keep a QKD system secure. An initial draft of the STIG is included in Appendix C.

Chapter 5

Conclusion

The HHUYS16 algorithm is a unique, new solution for trusted, third-party authenticated QKD. It is an algorithm that can be implemented using current technology with a sample reference implementation (Section 3.3) verified by experimentation (Section 3.1). With this algorithm, the confidentiality of existing QKD algorithms are preserved, and the integrity of authorization and validation is implemented in a sufficiently secure way.

The reference implementation has shown to be successful in a simulated BB84 execution environment. This simulation environment is verified and validated to operate in a way that simulates a quantum environment. The environment and reference implementation are made freely available as part of this dissertation to encourage repeatability of the results of HHUYS16.

Since the algorithm is written in a way that affords implementation on DoD networks, guidance and documentation are provided in the form of a QKD STIG (Appendix C). By meeting the requirements of this guide, a System Owner is able to implement a secure, trusted, third-party QKD solution. While the algorithm and contributions are complete, the DoD implementation requires future coordination with other organizations. A recommended solution is expounded upon in Section 4.2.1. In December 2015, the NSA announced that the Suite B cryptography solutions were being changed to allow and identify quantum-resistant algorithms. It's not clear when the FIPS-140-2 quantum-resistant algorithms will be permitted as compliant with DoD policy; however, the switch-over to such algorithms should start being implemented in development environments as part of a larger defense-in-depth strategy in preparation of a sudden switch by NIST and DoD.

Once the solutions and recommendations are implemented, the algorithm can be improved upon with the updates and improvements in Section 4.1. NTRU encryption is used as the example

extension method; however, any of the quantum-resistant, amended NSA Suite B cryptography algorithms and digital signature methods can be implemented in its place to further increase performance. The selection of very efficient encryption algorithms with large amounts of overhead can often lead to greater chances of entropy failure within the system.

Existing cryptography systems should not wait to implement quantum-resistant cryptography methods. While the specific algorithms permitted under NSA and DoD guidance may change, the process by which they are implemented will not change. Preparing for a massive shift in cryptographic requirements now will make the transition away from the classical encryption techniques easier to implement. An approach to encryption such as the HHUYS16 algorithm framework described in this dissertation foments a modular encryption design within larger systems and allows for more seamless upgrades in the future.

The HHUYS16 algorithm in its One-Time Pad form is implemented and shown to be complete (Section 3.3). It is usable now, and due to the provable nature of the One-Time Pad, will likely be permitted to continue as one of the NSA Suite B cryptography methods (even if the DoD does not implement the remaining recommendations). While an outline for using NTRU encryption is provided (resulting in performance improvements to the One-Time Pad implementation), there is no guarantee that NTRU will become part of and remain in the NSA Suite B collection. Until the future work is complete with the DoD, the One-Time Pad solution should be favored for implementing the HHUYS16 algorithm. A modular approach to selecting the encryption algorithm should be considered to allow adaptation and upgradeability as the QKD field is further developed by research.

The entirety of electronic communication is at risk. Encryption is not merely a task of outrunning the power behind a bruteforce attack: it's about outrunning the algorithms that thoroughly and completely break the encryption scheme altogether. HHUYS16 is complete and ready to be implemented now. It goes beyond the 1s and 0s of typical encryption and extends the quantum cryptography realm from merely confidential communiqué. It establishes that inherent

psychological need behind the secrecy and security in a way that no quantum cryptography algorithm has done before: it establishes trust. With that trust, a new level of integrity is created in the field of encryption, and HHUYS16 is the first step onto that level.

Bibliography

[1] A. D. Aczel, *Entanglement, The greatest mystery in physics*. Four Walls Eight Windows, 2001, ISBN: 1568582323.

[2] M. Ajtai, "Generating hard instances of lattice problems (extended abstract)," in *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96, Philadelphia, Pennsylvania, USA: ACM, 1996, pp. 99–108, ISBN: 0-89791-785-5. DOI: 10.1145/237814.237838. [Online]. Available: http://doi.acm.org/10.1145/237814.237838.

[3] "AR 25-2 - information assurance," Tech. Rep., Mar. 2009.

[4] P. W. Atkins, *Quanta: A Handbook of Concepts*, ser. Oxford chemistry series. Clarendon Press, 1974, ISBN: 9780198554936.

[5] O. Balci, "Verification validation and accreditation of simulation models," in *Proceedings of the 29th conference on Winter simulation*, IEEE Computer Society, 1997, pp. 135–141.

[6] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "NIST 800-57 - recommendation for key management – part 1: General (revision 3)," Tech. Rep., Jul. 2012.

[7] H. Bechmann-Pasquinucci and N. Gisin, "Incoherent and coherent eavesdropping in the six-state protocol of quantum cryptography," *Phys. Rev. A*, vol. 59, pp. 4238–4248, 6 Jun. 1999. DOI: 10.1103/PhysRevA.59.4238. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevA.59.4238.

[8] J. S. Bell, "On the Einstein Podolsky Rosen paradox," *Physics*, vol. 1, no. 3, pp. 195–200, 1964.

[9] C. H. Bennett, "Quantum cryptography using any two nonorthogonal states," *Phys. Rev. Lett.*, vol. 68, pp. 3121–3124, 21 May 1992. DOI: 10.1103/PhysRevLett.68.3121. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.68.3121.

[10] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," 150, IEEE, New York, Bangalore, India, 1984, pp. 175–182.

[11] C. H. Bennett, G. Brassard, and N. D. Mermin, "Quantum cryptography without Bell's theorem," *Phys. Rev. Lett.*, vol. 68, pp. 557–559, 5 Feb. 1992. DOI: 10.1103/PhysRevLett.68.557. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.68.557.

[12] A. Berthiaume, W. Van Dam, and S. Laplante, "Quantum Kolmogorov complexity," in *Computational Complexity, 2000. Proceedings. 15th Annual IEEE Conference on*, IEEE, 2000, pp. 240–249.

[13] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, "Post-quantum key exchange for the tls protocol from the ring learning with errors problem," IACR Cryptology ePrint Archive, 2014. http://eprint. iacr. org/2014/599. 3, 16, Tech. Rep., 2014.

[14] C. Branciard, N. Gisin, B. Kraus, and V. Scarani, "Security of two quantum cryptography protocols using the same four qubit states," *Physical Review A*, vol. 72, no. 3, p. 032 301, 2005.

[15] D. Bruss, "Optimal eavesdropping in quantum cryptography with six states," *Physical Review Letters*, vol. 81, no. 14, p. 3018, 1998.

[16] D. Bruss and C. Macchiavello, "Optimal eavesdropping in cryptography with three-dimensional quantum states," *Phys. Rev. Lett.*, vol. 88, p. 127 901, 12 Mar. 2002. DOI: 10.1103/PhysRevLett.88.127901. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.88.127901.

[17] A. Cabello, "Quantum key distribution in the holevo limit," *Phys. Rev. Lett.*, vol. 85, pp. 5635–5638, 26 Dec. 2000. DOI: 10.1103/PhysRevLett.85.5635. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.85.5635.

[18] "CJCSM 6510.01b - cyber incident handling program," Tech. Rep., Jul. 2012.

[19] "CJCSM 6510.01f - information assurance (ia) and support to computer network defense (cnd)," Tech. Rep., Feb. 2011.

[20] "CNSSI 1253 - security categorization and control selection for national security systems," Tech. Rep., Mar. 2014.

[21] J. Cohen, "Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit.," *Psychological Bulletin*, vol. 70, no. 4, pp. 213–220, 1968, ISSN: 0033-2909.

[22] *Defense Acquisition Guidebook 2013*. [Online]. Available: https://dag.dau.mil/.

[23] N. S. Dattani and N. Bryans, "Quantum factorization of 56153 with only 4 qubits," *ArXiv preprint arXiv:1411.6758*, 2014.

[24] T. A. Davidson, "Formal verification techniques using quantum process calculus," PhD thesis, University of Warwick, 2012.

[25] "DoDD 1322.18 - military training," Tech. Rep., Jan. 2009.

[26] "DoDI 3305.13 - DoD security education, training, and certification," Tech. Rep., Feb. 2014.

[27] "DoDI 5000.61 - DoD modeling and simulation (M&S) verification, validation, and accreditation (VV&A)," Tech. Rep., Dec. 2009.

[28] "DoDI 5200.1-r - information security program," Tech. Rep., Jan. 1997.

[29] "DoDI 5200.40 - DoD information technology security certification and accreditation process (DITSCAP)," Tech. Rep., Dec. 1997.

[30] "DoDI 8500.01e - information assurance (IA)," Tech. Rep., Oct. 2002.

[31] "DoDI 8500.2 - information assurance (IA) implementation," Tech. Rep., 2003, E3.

[32] "DoDI 8510.01 - DoD information assurance certification and accreditation process (DIACAP)," Tech. Rep., Nov. 1997.

[33] "DoDI 8510.01 - risk management framework (RMF) for DoD information technology (IT)," Tech. Rep., Mar. 2014.

[34] "DoDI 8551.01 - ports, protocols, and services management (PPSM)," Tech. Rep., May 2014.

[35] "DoDM 5200.01, volume 4 - DoD information security program: Controlled unclassified information (CUI)," Tech. Rep., Feb. 2012.

[36] L. Ducas and P. Q. Nguyen, "Learning a zonotope and more: Cryptanalysis of ntrusign countermeasures," in *Advances in Cryptology–ASIACRYPT 2012*, Springer, 2012, pp. 433–450.

[37] A. Einstein, B. Podolsky, and N. Rosen, "Can quantum-mechanical description of physical reality be considered complete?" *Physical review*, vol. 47, no. 10, p. 777, 1935.

[38] A. K. Ekert, "Quantum cryptography based on Bell's theorem," *Physical review letters*, vol. 67, no. 6, pp. 661–663, 1991.

[39] M. Elboukhari, M. Azizi, and A. Azizi, "Analysis of the security of BB84 by model checking," *International Journal of Network Security & Its Applications*, vol. 2, no. 2, 2010. DOI: 10.5121/ijnsa.2010.2207.

[40] D. G. Enzer, P. G. Hadley, R. J. Hughes, C. G. Peterson, and P. G. Kwiat, "Entangled-photon six-state quantum cryptography," *New Journal of Physics*, vol. 4, no. 1, p. 45, 2002.

[41] *FIPS pub 140-2, security requirements for cryptographic modules*, U.S.Department of Commerce/National Institute of Standards and Technology, 2002.

[42] S. Fluhrer, "Quantum cryptanalysis of NTRU," Cryptology ePrint Archive, Report 2015/676, http://eprint. iacr. org, Tech. Rep., 2015.

[43] S. J. Gay, R. Nagarajan, and N. Papanikolaou, "Qmc: A model checker for quantum systems," in *Computer Aided Verification*, Springer, 2008, pp. 543–547.

[44] C. Gentry, "A fully homomorphic encryption scheme," PhD thesis, Stanford University, 2009.

[45] D. C. Giancoli, *Physics for Scientists & Engineers with Modern Physics*, Third Edition. Prentice Hall, 2000, ISBN: 0130215171.

[46] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Rev. Mod. Phys.*, vol. 74, pp. 145–195, 1 Mar. 2002. DOI: 10.1103/RevModPhys.74.145. [Online]. Available: http://link.aps.org/doi/10.1103/RevModPhys.74.145.

[47] L. Goldenberg and L. Vaidman, "Quantum cryptography based on orthogonal states," *Physical Review Letters*, vol. 75, no. 7, p. 1239, 1995.

[48] B. Greene, *The Hidden Reality, Parallel Universes and the Deep Laws of the Cosmos.* Alfred A. Knopf, 2011, ISBN: 9780307265630.

[49] J. Gribbin, *Schrödinger's Kittens and the Search for Reality, Solving the Quantum Mysteries.* Little, Brown & Company, 1995, ISBN: 0316328383.

[50] R. B. Griffiths and C.-S. Niu, "Semiclassical Fourier transform for quantum computation," *Physical Review Letters*, vol. 76, no. 17, p. 3228, 1996.

[51] "Guide for developing security plans for federal information systems," Tech. Rep., Feb. 2006. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-18-Rev1/sp800-18-Rev1-final.pdf.

[52] N. Herbert, *Quantum Reality.* Anchor Books, 1987, ISBN: 0385235690.

[53] B. Huttner, N. Imoto, N. Gisin, and T. Mor, "Quantum cryptography with coherent states," *Physical Review A*, vol. 51, no. 3, p. 1863, 1995.

[54] Y. Kanamori, "Quantum encryption and authentication protocols," PhD thesis, University of Alabama in Huntsville, 2006, ISBN: 9780542805615.

[55] Y.-H. Kim, R. Yu, S. P. Kulik, Y. Shih, and M. O. Scully, "Delayed 'choice' quantum eraser," *Physical Review Letters*, vol. 84, no. 1, p. 1, 2000.

[56] M. Koashi and N. Imoto, "Quantum cryptography based on split transmission of one-bit information in two steps," *Phys. Rev. Lett.*, vol. 79, pp. 2383–2386, 12 Sep. 1997. DOI: 10.1103/PhysRevLett.79.2383. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.79.2383.

[57] S. Lloyd, *Programming the Universe, A Quantum Computer Scientist Takes On the Cosmos*. Alfred A. Knopf, 2006, ISBN: 1400033861.

[58] E. Martin-Lopez, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O'Brien, "Experimental realization of Shor's quantum factoring algorithm using qubit recycling," *Nature Photonics*, vol. 6, no. 11, pp. 773–776, 2012.

[59] "MIL-STD-3022 - documentation of verification, validation, and accreditation (VV&A) for models and simulations," Tech. Rep., Jan. 2008.

[60] T. Miyadera and H. Imai, "Quantum Kolmogorov complexity and quantum key distribution," *Physical Review A*, vol. 79, no. 1, p. 012 324, 2009.

[61] E. Okada, R. Tanburn, and N. S. Dattani, "Reducing multi-qubit interactions in adiabatic quantum computation without adding auxiliary qubits. part 2: The" split-reduc" method and its application to quantum determination of ramsey numbers," *ArXiv preprint arXiv:1508.07190*, 2015.

[62] R. A. Perlner and D. A. Cooper, "Quantum resistant public key cryptography: A survey," in *Proceedings of the 8th Symposium on Identity and Trust on the Internet*, ACM, 2009, pp. 85–93.

[63] J. M. Pollard, "A monte carlo method for factorization," *BIT Numerical Mathematics*, vol. 15, no. 3, pp. 331–334, 1975.

[64] J. Proos and C. Zalka, "Shor's discrete logarithm quantum algorithm for elliptic curves," *ArXiv preprint quant-ph/0301141*, 2003.

[65] H. Riesel, *Prime numbers and computer methods for factorization*. Springer Science & Business Media, 2012, vol. 126.

[66] F. E. Ritter, M. J. Schoelles, K. S. Quigley, and L. C. Klein, "Determining the number of simulation runs: Treating simulations as theories by not sampling their behavior," in *Human-in-the-Loop Simulations*, Springer, 2011, pp. 97–116.

[67] G. Rose and W. Macready, "An introduction to quantum annealing," *D-Wave Systems*, 2007.

[68] R. G. Sargent, "Verification and validation of simulation models," *Journal of simulation*, vol. 7, no. 1, pp. 12–24, 2013.

[69] V. Scarani, A. Acín, G. Ribordy, and N. Gisin, "Quantum cryptography protocols robust against photon number splitting attacks for weak laser pulse implementations," *Phys. Rev. Lett.*, vol. 92, p. 057 901, 5 Feb. 2004. DOI: 10.1103/PhysRevLett.92.057901. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.92.057901.

[70] B. Schumacher, "Information from quantum measurements," in *Complexity, Entropy and the Physics of Information (Santa Fe Institute Studies in the Sciences of Complexity Proceedings)*, W. H. Żurek, Ed., Perseus Books (Sd), 1989, pp. 29–38, ISBN: 0201515091.

[71] C. Shannon and W. Weaver, *The Mathematical Theory of Communication*. University of Illinois Press, 1949, ISBN: 9780252725463.

[72] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM journal on computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[73] M. Swanson, J. Hash, and P. Bowen, "Security and privacy controls for federal information systems and organizations," Tech. Rep., Apr. 2013. DOI: 10.6028/nist.sp.800-53r4. [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-53r4.

[74] K. Tamaki and H.-K. Lo, "Unconditionally secure key distillation from multi-photons," *Phys. Rev. A*, vol. 73, p. 010 302, 1 Jan. 2006. DOI: 10.1103/PhysRevA.73.010302. eprint: arXiv:quant-ph/0412035.

[75] V. Vedral, M. B. Plenio, M. A. Rippin, and P. L. Knight, "Quantifying entanglement," *Physical Review Letters*, vol. 78, no. 12, p. 2275, 1997.

[76] Wikipedia, *Mach–zehnder interferometer — wikipedia, the free encyclopedia*, [Online], 2014. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Mach%E2%80%93Zehnder_interferometer&oldid=597131476.

[77] ——, *Entropy (information theory) — wikipedia, the free encyclopedia*, [Online], 2016. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Entropy_(information_theory)&oldid=720464539.

[78] T. Young, *A course of lectures on natural philosophy and the mechanical arts.* London: Printed for J. Johnson, 1802, vol. 1. [Online]. Available: http://www.biodiversitylibrary.org/item/63005.

Appendices

Appendix A

Failed and Inefficient Algorithms

The algorithms presented in this dissertation have undergone several years of development, and some iterations of the algorithm failed to pass the cryptanalysis stage. Some of the failed algorithms that present logical and cryptanalytic problems are presented here. These problems were usually manifest the same way in different locations of the algorithm.

A.1 Unsecured Encryption Attack

1. A sends B a message on the classical channel that she would like to initiate a trusted quantum key distribution.

2. A sends B a series of entangled photons that will become the key.

3. A then announces to T on the classical channel that she would like to communicate securely.

4. T sends A photons on the quantum channel.

5. A and T negotiate the measurement basis as in the BB84 algorithm. The photons sent in step 4 become a temporary pad, P_A, which will be used to pad future messages. P_A will be divided into thirds, P_A^1, P_A^2, and P_A^3.

6. A prepares a message, M_A, for T stating that she would like to communicate with B. This message is encrypted with K_A. She sends this message to T on the classical channel: $\epsilon(K_A, M_A)$.

7. ...

In this iteration of the algorithm, the message sent from A to T is encrypted with the private key negotiated during the registration phase of A to T. Assuming that the encryption algorithm is a quantum-resistant algorithm such as NTRU, it has a limited lifespan of usefulness and indeterminable number of uses before enough information is leaked to determine the lattices used for encryption.

Herein lies one of the key features of the encryption portion of this thesis' main algorithm: the encryption piece does not need to be abundantly secure. It is used as a verification measure — a digital signature — rather than actual encryption. Using a strong method of encryption does provide an increased defense-in-depth strategy for securing the communication, but any digital signature function would not severely decrease the security of the system at this point.

At this failure, it was deemed best to assume that the underlying encryption algorithm is insecure. This increases the usefulness and security of the overall algorithm.

A.2 Additional Hardware Requirements

1. A sends B a message on the classical channel that she would like to initiate a trusted quantum key distribution.

2. A sends B a series of entangled photons that will become the key.

3. A then announces to T on the classical channel that she would like to communicate securely.

4. T sends A photons on the quantum channel.

5. A and T negotiate the measurement basis as in the BB84 algorithm. The photons sent in step 4 become a temporary pad, P_A, which will be used to pad future messages. P_A will be divided into thirds, P_A^1, P_A^2, and P_A^3.

6. A prepares a message, M_A, for T stating that she would like to communicate with B. This message is first encrypted with K_A and then padded with the first 1/3 of P_A, which is P_A^1. She sends this message to T on the classical channel: $\phi(P_A^1, \epsilon(K_A, M_A))$.

7. T unpads and then decrypts the message
$(\delta(K_A, \upsilon(P_A^1, \phi(P_A^1, \epsilon(K_A, M_A)))) = M_A)$, then prepares to contact B on behalf of A. T announces on the classical channel to B that he would like to communicate securely.

8. B sends photons on the quantum channel to T.

9. B and T negotiate the measurement basis as in the BB84 algorithm. The photons sent in step 8 become a temporary pad, P_B, which will be used to pad future messages. P_B will be divided into thirds, P_B^1, P_B^2, and P_B^3.

10. B then prepares a message, M_B containing the measurement basis he used to read A's photons in step 2. M_B is first encrypted with K_B, then padded with the first 1/3 of P_B, which is P_B^1. This message is sent to T on the classical channel: $\phi(P_B^1, \epsilon(K_B, M_B))$.

11. T unpads the message containing the measurement basis from B (see step 10) then decrypts it to get M_B using the formula $(\delta(K_B, \upsilon(P_B^1, \phi(P_B^1, \epsilon(K_B, M_B)))) = M_B)$. He then pads it with P_B^3, encrypts it with K_A, then pads it again with P_A^2. This message,
$\phi(P_A^2, \epsilon(K_A, \phi(P_B^3, M_B)))$, is sent to A on the classical channel. A then unpads and decrypts this, and is left with $\phi(P_B^3, M_B)$ such that
$\delta(K_A, \upsilon(P_A^2, \phi(P_A^2, \epsilon(K_A, \phi(P_B^3, M_B))))) = \phi(P_B^3, M_B)$.

12. T prepares a message, T_B, with contains the last 1/3 of P_A (which is P_A^3), encrypts it with K_B, and pads it with P_B^2. T sends B the pad P_A^3 over the classical channel as $\phi(P_B^2, \epsilon(K_B, P_A^3))$, which B unpads and decrypts to obtain P_A^3: $\delta(K_B, \upsilon(P_B^2, \phi(P_B^2, \epsilon(K_B, P_A^3)))) = P_A^3$.

13. B pads P_B^3 with the result of step 12, P_A^3, and sends them to A on the classical channel: $\phi(P_A^3, P_B^3)$.

14. A unpads this communication from B to obtain
P_B^3: $\upsilon(P_A^3, \phi(P_A^3, P_B^3)) = P_B^3$. A then verifies B's identity through T by performing $\upsilon(P_B^3, \phi(P_B^3, M_B))$ (remember that A has $\phi(P_B^3, M_B)$ from step 11) which will result in the measurement basis

M_B. She then completes the negotiation stage of BB84 with B using the measurement basis in M_B.

In this iteration of the algorithm, the quantum key negotiation phase requires B to have quanta sending equipment. This equipment is expensive, so changing the quanta generation requirements to be absorbed by the sender of the data and the trusted third party decreases the costs of also having the recipient contain quanta generation hardware.

This version of the algorithm has the advantage of not requiring T to have quanta sending equipment. This decreases the cost of T while increasing the cost of every party A may need to communicate with.

Suppose, for example, that a central unit controls hundreds of fielded sensors. The central unit wants to check in and uses the algorithm in Section 3.2 to do so. In this case, none of the fielded sensors must have the quanta sending equipment.

This also provides a tactical advantage: the chosen implementation of the algorithm does not require bulky quanta generation equipment, meaning there is less hardware to supply in deployed tactical units.

Appendix B

Algorithm Attempts Per Entropy Failure

Qubits of Entropy	*Average Attempts per Failure*	*Average Attempts per Failure on 1 of 3 legs*
28	11356	3785
29	16060	5353
30	22713	7571
31	32121	10707
32	45426	15142
33	64242	21414
34	90852	30284
35	128484	42828
36	181704	60568
37	256968	85656
38	363408	121136
39	513937	171312
40	726817	242272
41	1027875	342625
42	1453635	484545
43	2055750	685250
44	2907270	969090
45	4111500	1370500
46	5814540	1938180
47	8223001	2741000
48	11629080	3876360

49	16446002	5482001
50	23258160	7752720
51	32892005	10964002
52	46516320	15505440
53	65784010	21928003
54	93032639	31010880
55	131568020	43856007
56	186065279	62021760
57	263136046	87712015
58	372130559	124043520
59	526272092	175424031
60	744261118	248087039
61	1052544096	350848032
62	1488522236	496174079
63	2105088192	701696064
64	2977044471	992348157
65	4210177804	1403392601
66	5954088943	1984696314
67	8420355609	2806785203
68	11908177887	3969392629
69	16840688505	5613562835
70	23816355774	7938785258
71	33681377010	11227125670
72	47632711549	15877570516
73	67362754021	22454251340
74	95265423098	31755141033
75	134725508041	44908502680

76	190530846196	63510282065
77	269456830736	89818943579
78	381061692393	127020564131
79	538913661473	179637887158
80	762123384785	254041128262
81	1077734294522	359244764841
82	1524246769571	508082256524
83	2155840734863	718613578288
84	3048493539143	1016164513048
85	4311681469727	1437227156576
86	6096987078286	2032329026095
87	8623362939455	2874454313152
88	12193974156573	4064658052191
89	17246725878910	5748908626303
90	24387948313146	8129316104382
91	34493451757819	11497817252606
92	48775896626292	16258632208764
93	68607854595223	22869284865075
94	97551793252583	32517264417528
95	138740328181452	46246776060484
96	195103586505167	65034528835056
97	271448468181102	90482822727034
98	390207173010335	130069057670112
99	567574069833214	189191356611071
100	780414346020669	260138115340223

Appendix C

Quantum Key Distribution Security Technical Implementation Guide Version 1 Release 0 (prerelease)

Developed by Jon Hood for release to all. *NOTE: Most STIGs are for use by the DoD. As such, the releasability of this STIG includes all entities.* Some language is standard documentation copied from other STIG guidance. This duplication is understood by DISA and should not be interpreted as an attempt at plagiarism.

C.1 General Changes

This is the first release of this STIG. This section is a placeholder for future revisions.

C.2 Introduction

C.2.1 Background

This Quantum Key Distribution Security Technical Implementation Guide (STIG) provides security guidance for use when generating cryptographically-secure keys for use in identification and encryption. This STIG provides the guidance needed to promote the implementation, use, and protection of a QKD subsystem. As vulnerabilities detailed here can require significant architectural changes to achieve, all systems are encouraged to implement these guidelines as early as possible. Significant effort may be required to implement these guidelines, and implementing them in the early stages of a QKD subsystem can be less disruptive to any potential remediation process.

C.2.2 Authority

This STIG is not yet authorized by any DoD organization. This section is a placeholder for when such authorization is given.

C.2.3 Scope

This document is not yet a requirement. It is intended for all DoD developed, architected, and administered enterprise applications and systems connected to DoD networks that implement QKD. These requirements are not intended to be applied to non-quantum networks, nor are they required replacements for other key distribution schemes.

C.2.4 Writing Conventions

NOTE: taken from existing STIGs

Throughout this document, statements are written using words such as "**will**" and "**should**." The following paragraphs are intended to clarify how these STIG statements are to be interpreted. A reference that uses "will" indicates mandatory compliance. All requirements of this kind will also be documented in the italicized policy statements in bullet format, which follow the topic paragraph. This makes all "**will**" statements easier to locate and interpret from the context of the topic. The IAO will adhere to the instructions as written.

For each italicized policy bullet, the text will be preceded by parentheses containing the STIG Identifier (STIGID), which corresponds to an item on the checklist and the severity code of the bulleted item. An example of this will be as follows: "(*G111: CAT II*)." Throughout the document, accountability is directed to the IAO to "ensure" a task is carried out or monitored. These tasks may be carried out by the IAO or delegated to someone else as a responsibility or duty.

A reference to "**should**" indicates a recommendation that further enhances the security posture of the site. These recommended actions will be documented in the text paragraphs but not in the italicized policy bullets. All reasonable attempts to meet this criterion will be made.

C.2.5 Vulnerability Severity Category Code Definitions

NOTE: taken from existing SCA-V definition on NETCOM SCA-V site

Severity Category Codes (referred to as CAT) are a measure of vulnerabilities used to assess a facility or system security posture. Each security policy specified in this document is assigned a Severity Code of CAT I, II, or III.

	DISA Category Code Guidelines
CAT I	Any vulnerability, the exploitation of which will, **directly and immediately** result in loss of Confidentiality, Integrity, or Availability.
CAT II	Any vulnerability, the exploitation of which **has a potential** to result in loss of Confidentiality, Integrity, or Availability.
CAT III	Any vulnerability, the existence of which **degrades measures** to protect against loss of Confidentiality, Integrity, or Availability.

Figure C.1: Vulnerability Severity Category Code Definitions

C.2.6 STIG Distribution

This STIG is available from Jon Hood's personal website and is authorized for redistribution in all formats. The STIG is available at the Uniform Resource Locator (URL) `http://www.hoodsecurity.com/Jon/`.

C.2.7 Document Revisions

Comments or proposed revisions to this document should be filed via the bugtracking system available at `https://www.hoodsecurity.com/Jon/bugzilla/`. A bugzilla project has been set up containing components for this document and for the associated XCCDF file.

C.2.8 Document Overview

This guide is intended to assist in the design, development, implementation, and maintenance of a QKD-enabled network. Security guidance regarding the hardening of operating systems and applications causes potential attackers to find weak points in the setup of those applications. There exists several methods of attempting to compromise a QKD scheme demonstrating the need for QKD best practices.

This guide is not specific to a particular QKD algorithm or method. As such, portions of this guide may not apply to all QKD subsystems. In some cases, guidance relates to a particular QKD method (IE: entanglement vs. non-entanglement algorithms). The presence of specific guidance for the system that is being used does not exempt other requirements against it. This guide should be applied to both internally-developed QKD schemes and third-party QKD implementations.

When using this guide to secure a third-party QKD implementation, specific technical details about the third-party product offering must be known. Being unable to verify or determine the internal workings of a commercial or third-party solution does not mean the checks are not-applicable; the checklist items are considered failed when they cannot be verified.

For best results, the guidance in this STIG should be implemented as early as possible in the QKD subsystem implementation process. When developing QKD-based solutions, third-party developers are encouraged to integrate the procedures listed here. Third-party developers are encouraged to create their own guidelines. This STIG, combined with the third-party security guidelines, forms the foundation of a secure configuration for implementation in a National Security System (NSS).

C.2.9 Document Organization

This document details the minimum configuration requirements for a QKD system, organized as follows:

- Section C.2 provides document overview and background information.

- Section C.3 provides project management requirements. The Project Manager role should take the responsibility for implementing the guidance in this section.
- Section C.4 details on the design and development of the QKD system. The development and technical teams should be responsible for integrating these requirements.
- Section C.5 deals with testing of the solution. A Test Manager or dedicated testing team should be responsible for verifying the solution.
- Section C.6 explains the deployment requirements of the QKD solution. Ultimately, the IAO (Information Assurance Officer) is responsible for verifying the security here and at all prior levels.

C.3 Project Management

This section describes the IA responsibilities of the Project Manager (PM). The PM's first responsibility is to follow the guidance of the Defense Acquisition Guidebook (DAG), particularly Chapter 11.[22] The DAG is currently being rewritten to follow RMF control requirements. Second, the PM should verify that all RMF controls in the PM-* family are implemented, available in Appendix G of the NIST 800-53r4 documentation.[73]

C.3.1 Documentation

The PM is responsible for verifying and disseminating the System Security Plan (SSP) and Application Configuration Guide to other team members. It is the PM's responsibility to make sure that these documents stay updated, and that everyone is following them.

System Security Plan (SSP)

The PM is required to produce a SSP. These documents provide an overview of the security requirements of the application. The SSP must contain:

- Technical, Administrative, and Procedural IA program policies
- Designation of IA personnel
- Identification of IA requirements
- IA objectives

Also, there must be supplementary documentation (which can be included in the main SSP documentation) detailing:

- Appointments to IA roles
- Assigned duties
- Appointment criteria (ie: training, clearance, certifications)

QKD2016.1. ***CAT II*** The Program Manager will ensure that a SSP is established documenting the technical, administrative, and procedural IA program and policies governing the DoD information system. All IA personnel and specific IA requirements and objectives must be documented. The SSP must explicitly include the QKD system and training required to operate the system within this plan.[51]

Discussion A complete SSP must include the important IA assets and point the reader to a location that can further define the technical details of the IA system. Without explicitly documenting the QKD system in the SSP, further risks to duty rotation, continuity of operation planning (COOP), and insider threats are increased.

Check Contents Ask the system representative for a copy of the SSP. Verify that the QKD system is specifically listed as part of the plan. Verify that any IA roles are properly defined.

IA Controls PL-2, CCI-003050

QKD2016.2. ***CAT II*** The Program Manager will ensure all appointments to IA roles are established in writing, including assigned duties and appointment criteria.

Discussion Appointment letters, proper certification, and other IA criteria are required at certain IA roles. Without proper training, an individual on the system may inadvertently circumvent the security controls that are in place.

Check Contents Ask the system representative for a copy of the appointment letters, training status, and certification of individuals within SSP-defined IA roles. If no IA roles are defined, this is a finding. If appointment letters, training status, or certification is expired or not available, this is a finding.

IA Controls PL-2, CCI-003059

QKD Configuration Guide

The PM is responsible for creating a QKD Configuration Guide detailing the secure deployment process for the subsystem. This information is needed by all providers that need to connect (Authority To Connect requirements, DD1144 MOAs, etc.). The PM is the liaison between third-parties and the main system. The PM should be able to present a third-party with the product and the configuration guide, and the third-party should have sufficient information for setting up the instance of the QKD subsystem in a secure manner.

When the categorization of a system has high Confidentiality, the QKD subsystem will not be hosted by or share resources with components being used for other purposes. When the categorization of a system has high Integrity, an authentication mechanism, such as HHUYS16, will also be implemented.

The PM will detail the categorization requirements in the configuration guide. Any outside connections will be documented with proper rules and requirements as defined by the organization.

QKD2016.3. ***CAT II*** A QKD Configuration Guide will be available internally and as part of any third-party interconnection requirements.

Discussion External parties must be able to duplicate the QKD subsystem. Scenarios, including physical damage to the system or loss of critical personnel on the system, require that others be able to step in and securely install, configure, and maintain the system. A configuration guide for getting the system into an operational, secure state must be available.

Check Contents Ask the system representative for the QKD Configuration Guide. If the configuration guide does not exist, or if the configuration guide does not include steps for setting up the QKD subsystem, this is a finding.

IA Controls CM-6, CCI-000366

QKD2016.4. ***CAT III*** A QKD Configuration Guide will list connection rules from third-parties.

Discussion External parties that are attempting to authorize their connection to the quantum network must be able to comply with the hosting system's requirements.

Check Contents If this QKD instance is on a closed, restricted network without access to an externally connecting system, this check is NA. Ask the system representative for the QKD Configuration Guide. If the configuration guide does not exist, or if the configuration guide does not include rules for establishing third-party connections, this is a finding.

IA Controls AC-4, CCI-001549

QKD2016.5. ***CAT III*** The system will provide documentation of interconnected subsystems.

Discussion External parties that are attempting to authorize their connection to the quantum network must be able to comply with the hosting system's requirements. When they comply, their acceptance and compliance with the requirements must be stored by the hosting system.

Check Contents If this QKD instance is on a closed, restricted network without access to an externally connecting system, this check is NA. Ask the system representative for the QKD Configuration Guide. If connecting systems have not complied with the requirements of the configuration guide, this is a finding.

IA Controls AC-4, CCI-001414

QKD2016.6. ***CAT II*** The system will define baseline configuration options and development environment requirements.

Discussion Often, development environments do not match the production environments documented in the QKD Configuration Guide. The configuration guide should therefore define the additional requirements in setting up a separate development environment for the QKD system.

Check Contents Ask the system representative for the QKD Configuration Guide. If the guide does not exist or if a sandbox or development environment is not defined in the configuration guide, this is a finding.

IA Controls SA-4, CCI-003099

Security Classification Guide

The PM creates and updates the Security Classification Guide for the system in accordance with DoD 5200.1-R.[28] This duty can be shared with the IAO/IAM.

QKD2016.7. *CAT II* A Security Classification Guide will be created.

Discussion Different parts of a system can handle data at different levels of classification. Which parts are allowed to handle with data must be defined. Users must be able to quickly know the classification of data they are looking at. This guide avoids classified and unclassified material from being mixed, exposing the system to possible data leakage events.

Check Contents If the system does not handle classified data, this check is NA. Ask the system representative for a copy of the Security Classification Guide. If a guide does not exist, this is a finding. If the elements of the QKD system handle classified data or are used to encrypt classified data, these elements must be defined in the guide. If they are not, this is a finding.

IA Controls CA-3, CCCI-000260

C.3.2 Training

All levels of personnel require degrees of training to properly safeguard a system. Upper management must have full buy-in to the security policy of the system, and must be aware of the overall security goals the system is attempting to accomplish. Engineers and developers must

be familiar with secure design and development principles to avoid taking shortcuts, introducing unauthorized third-party components, and exposing the system to side-channel attacks outside of the security protections of the system. Test engineers must understand the nature of data they are recording and protect such data.

QKD2016.8. ***CAT II*** All development staff receive proper training.

Discussion Well-trained development personnel are the first line of defense in an in-depth security policy. Lack of sufficient training leads to oversights, compromise, and loss of system Confidentiality, Integrity, or Availability.[25]

Check Contents Obtain the training and certification data for developers of the QKD system. If training programs do not exist or the developers have not maintained certification in the programs, this is a finding.

IA Controls SA-16 CCI-003292

QKD2016.9. ***CAT II*** All staff receive proper training.

Discussion Well-trained personnel are the first line of defense in an in-depth security policy. Lack of sufficient training leads to oversights, compromise, and loss of system Confidentiality, Integrity, or Availability.[26]

Check Contents Obtain the training and certification data for personnel with access to the QKD system. If training programs do not exist or the developers have not maintained certification in the programs, this is a finding.

IA Controls PM-14, CCI-003000

C.3.3 Maintenance

Security Incident Response Process

Preparing for a security-related event requires that a standard operating procedure has been established. This procedure defines what types of events to look for, who is responsible for looking for those events, what to do when a security-related event is triggered, and who is responsible with analysis and follow-up action items from the event. A Computer Network Defense Service Provider is often used to monitor a network for such malicious events. In a quantum network, there is an additional set of events to monitor for.

The nature of QKD provides for a way of detecting eavesdroppers; what is required when an eavesdropper is detected? How are the threats removed? What happens when communication pathways are blocked between quantum nodes? A detailed incident response plan allows management to handle such events.[18]

QKD2016.10. ***CAT II*** A detailed incident response plan is in place.

Discussion Without planning, training, and exercising incident response plans, personnel will not know how to look for and report potentially harmful events on the system.

Check Contents Verify that an incident response program and policy is in place. Make sure that personnel are aware of who to go to if they have questions regarding the incident response capability. If a CNDSP team is used, verify that they have expertise in quantum networks. If no documentation exists that details the incident response procedures, training for the incident response team, and a timeline on when the procedures are exercised, this is a finding.

IA Controls CP-2, CCI-000460; IR-1, CCI-000805

Vulnerability Management

When security updates are made available to the QKD components, how are these updates obtained by the system? Timely updates of the system protect from security issues. The system must use a supported, fully-patched version of the components from the vendor.[30, 19]

QKD2016.11. ***CAT II*** The Program Manager will ensure that updates are applied in a timely manner.

Discussion When the vendor supplies updates to the QKD system, the updates must be installed. At a minimum, the updates should be included in the system IAVM processes. Quickly patching the system with a supported version of the components helps ward off new attacks and zero-day vulnerabilities.

Check Contents If updates are not made available to the QKD components, this is a finding. If the update repositories are not subscribed to or included in a system IAVM process, this is a finding. If updates are available for a component and the update changelog includes security-related updates, this is upgraded to a CAT I finding.

IA Controls SI-2, CCI-002605; SI-2, CCI-002607

Security flaws are often not discovered by the vendor. Users and other personnel must be made aware of how to report potential security issues within the system. The Program Manager is responsible for creating a comprehensive vulnerability management process, including notification to users, for when issues are discovered.

QKD2016.12. ***CAT II*** The Program Manager will ensure that a comprehensive vulnerability management strategy is established.

Discussion A comprehensive vulnerability management process, when created and implemented in the design phase of the system, promotes visibility and accountability through the lifecycle of the system. Response to security events are more streamlined and timely when responding to potential threats.

Check Contents Verify that a comprehensive vulnerability management process exists and is documented. The process must include:

1. Users of the system must be notified of potential breaches of information.
2. Any personnel involved with the system must be made aware of whom to report potential security issues to.
3. A triage timeline for fixing security events must be established, not to exceed 30 days.

If a comprehensive vulnerability management process does not exist or is not complete, this is a finding.

C.3.4 Workplace Procedures

The Program Manager is in charge of developing procedures that ensure the Confidentiality, Integrity, and Availability of sensitive data. Data must be physically handled properly, including checkpoints that verify data storage. The policy is unique per system depending on the sensitivity of the data it handles.[35]

QKD2016.13. ***CAT II*** The Program Manager ensures that a data management policy is consistent in assuring protection at the data's sensitivity level.

Discussion A procedure must exist that defines how data are handled through the system. This policy must cover, if deemed appropriate at the sensitivity level:

1. data in-transit encryption requirements

2. data at-rest protections

3. end-of-day and random security checks of data

4. two-person handling rules

Check Contents Determine the sensitivity of the data handled by the system. If data sensitivity is higher than fully unclassified without caveats, ask for the data management and handling policy that addresses handling and retention of data. If data are output from the system, verify that there is a policy that data must be marked with its proper sensitivity level.

If data are classified, verify that policy addresses end-of-day checks and requires random checks, including SF 701 and SF 702 checklist sheets.

If guidance does not address required information or is not being followed, this is a finding.

IA Controls SI-12, CCI-001315; SI-12, CCI-001678

QKD2016.14. ***CAT II*** Data sensitivity must be properly marked on all media.

Discussion All media handled by the system must be marked at its proper sensitivity level. This prevents inadvertent data leakage, malicious attempts to circumvent higher classification security controls, and encourages handlers to use proper methods when in charge of the data.

Check Contents Verify that output media, including backups, printouts, and recovery tools, are marked with their proper sensitivity level. Verify that DoDM 5200.01 M Vol. 1-4 requirements are implemented.

IA Controls MP-3, CCI-001010

QKD2016.15. ***CAT II*** The system maintains data protections for data leaving the boundaries of the system.

Discussion All media handled by outside systems must preserve the safeguards present on the data.

Check Contents Verify that data are handled according to DoDM 5200.01 M Vol. 1-4 and DoDD 5015.2.

IA Controls MP-5, CCI-001020

C.3.5 DoD Standards Compliance

Often, the QKD system is composed of several other systems. These subsystems must also be compliant with their respective STIGs, NSA Guidance, manufacturer hardening guides, and any other DoD policies. The Program Manager should designate a security lead in charge of DoD compliance.

QKD2016.16. ***CAT II*** The Program Manager ensures compliance with DoD policies.

Discussion The Program Manager will ensure that DoD STIGs, NSA guides, manufacturer hardening requirements, and any other DoD policy is applied to all components of the underlying system. For example, if the QKD system relies on an embedded RedHat operating system, the embedded operating system must comply with the RedHat STIG. DoD STIGs are available from `http://iase.disa.mil/stigs/index.html`. NSA guides are available at `http://www.nsa.gov/ia/guidance/index.shtml`.

Check Contents Verify any components of the QKD system that have DoD policies for them. If these policies are not applied, including additional STIGs, NSA guides, manufacturer hardening guides, and any other DoD policy, this is a finding.

IA Controls CM-6, CCI-000366

C.4 Design and Development

Security elements must be factored into the entire lifecycle of a system. QKD components, when designed with security in mind, cost less to comply with policy. Guidelines for further development and documentation of the system are easy to implement when the foundation for these security policies is implemented.

C.4.1 Documentation

Design Document

The designer of the system is primarily responsible with developing and updating the System Design Document. This document covers many aspects of the system design and references the minimum security requirements for it. Any essential functions must also be documented for continuity of operations requirements in this document as well.

QKD2016.17. ***CAT II*** The Designer will create and update a Design Document for the QKD system.

Discussion An appropriate design document allows a competent third party to understand how the system works and what is required to run the system in a secure manner.

Check Contents Ask the system representative for the Design Document. Verify that the Design Document contains the following:

- list of all external interfaces
- nature of data exchanged
- categories of data sensitivity (EG: FOUO, PII, HIPAA)
- protections on each interface
- users roles
- access privileges and requirements to each role
- restoration priority of QKD subcomponents

IA Controls PL-8, CCI-003073

Configuration Guide

A QKD Configuration Guide is created during development to detail the best practices and secure configuration of the QKD system. Any special configuration required to operate the system must be documented, and the guide should make recommendations for secure default options.

If a configuration option can affect the security posture of the system, it must be documented with the relevant security concerns. Any risks that can be attributed to the security configuration of the system should be accompanied with:

- precautions
- potential risk mitigations
- recommendations and reasoning behind selected defaults

QKD2016.18. ***CAT II*** A QKD Configuration Guide will detail the setup of a secure implementation of the QKD system.

Discussion A QKD Configuration Guide can help system engineers configure the system in a secure way. Recovery of mission-critical assets can be streamlined with a good configuration guide. The configuration guide should also be stored at the COOP location.

Check Contents Verify that a QKD Configuration Guide exists. If the document does not contain enough information to set up the QKD system into a working state, this is a finding. If the configuration recommendations in the document do not produce a secure installation of the QKD system, this is a finding.

IA Controls SA-4(3), CCI-003108

QKD2016.19. ***CAT II*** The system's configuration is documented in the QKD Configuration Guide

Discussion The existence of a configuration guide shows that it can be configured in a secure way. The system itself must be documented in the guide.

Check Contents Verify that the system's configuration is documented in the QKD Configuration Guide. If the configuration of the system differs from the recommendations in the configuration guide, those differences and justifications are recorded in the guide. If they are not, or if the QKD Configuration Guide does not exist, this is a finding.

IA Controls SA-4(3), CCI-003108

Threat Modeling

Each QKD implementation is different. Satellite systems have differing physical and environmental security concerns than an implementation on a closed, restricted network guarded by soldiers. Threat models should be performed for the potential deployment. Mitigations to threats and vulnerabilities should be recommended based on the unique characteristics of the deployment.

QKD2016.20. ***CAT II*** Threat models for the configuration must be analyzed.

Discussion For the QKD implementation, known threats should be modeled and mitigated.

Check Contents Verify that threat models exist for the QKD implementation. If they do not exist, this is a finding. If the threats do not include risk analysis, this is a finding. If the risk analysis can be mitigated and the mitigations are not performed by the instance, this is a finding.

IA Controls SA-15(4), CCI-003256

Threat modeling can be performed in several stages. While the specific type of threat modeling may be unique to the QKD implementation, an option for performing the modeling is outlined:

1. **Define Common Usage** — The common usage criteria should be defined. This is the default success scenario and intended purposes of the QKD subsystem.

2. **Identify External Dependencies** — Often, the QKD subsystem will use a shared source of entangled quanta. These dependencies and required components should be well-defined. Their implementation should be managed at a similar level to the classification of the QKD system.

3. **Enumerate Security Assumptions** — The specific defaults and allowed algorithms should be defined here, as well as assumptions for their implementation. The identification of a trusted third-party should also be one of the assumptions here.

4. **Identify Interactions** — What medium is being used for the classical channels? Is it a network managed at the same classification level as the QKD subsystem? Interactions with other components and the requirements for those interactions are documented here.

5. **Identify Entry Points** — A quanta reader will be needed for measuring the received quanta. Additionally, the protocols and methods of access on the classical channel should be enumerated here. Each entry point discovered should be enumerated separately with individual threat analysis.

6. **Risk Calculation** — A risk calculation, taking into account the impact and likelihood of the threats to the entry points, should be performed.

Ports, Protocols, and Services Management

A Ports, Protocols, and Services Management document should be created detailing all provided and utilized communication pathways.[34]

QKD2016.21. ***CAT II*** Ports, Protocols, and Services Management

Discussion Identifying which ports and protocols are used allows side and covert channel attacks to be more easily detected. The PPSM document details the ports, APIs, and any communication pathways in the system. This should be included in a data flow diagram with explicitly permitted communication methodology detailed in a PPSM document.

Check Contents Ask the system representative if a PPSM document exists. If it does not exist, or if communication occurs outside of the explicitly design PPSM pathways, this is a finding.

IA Controls CM-7(3), CCI-000388

QKD2016.22. ***CAT II*** Data Flow Diagram

Discussion Identifying which interfaces can interface with others is part of the Data Flow Diagram. This diagram details the validation boundaries and whitelisted communication paths. Only ports in the PPSM document should be allowed.

Check Contents Ask the system representative for a data flow diagram or system boundary diagram. If the communication interfaces communicate in ways not authorized by the PPSM document, this is a finding. If either a data flow diagram or PPSM document do not exist, this is a finding.

IA Controls CA-2, CCI-000248

C.4.2 Third-Party Tools

The QKD implementation often relies on additional GOTS and COTS components. These components should be evaluated for security during the design and development stages of the QKD subsystem. It is the PM's and Developer's jobs to make sure that the components used in the subsystem comply with all the requirements.[3]

QKD2016.23. ***CAT III*** All third-party components must be approved for use on the system.

Discussion For the QKD implementation, all third-party products must be evaluated and approved prior to implementation.

Check Contents Verify that all COTS, GOTS, and other third-party components have been evaluated by either the NIAP approval process (see `http://www.niap-ccevs.org/cc-scheme/vpl/`http://www.niap-ccevs.org/cc-scheme/vpl/) or at least an Assess-Only RMF validation in NIST 800-53. If the components are not approved, are approved but configured in an unapproved way, or implement unapproved protocols (IE: non-FIPS encryption algorithms), this is a finding.

IA Controls SA-4(7), CCI-000634

C.4.3 Best Practices

Following several best practices in QKD implementation allows for a consistent quality in each instance. Certain practices provide a defense-in-depth approach that enhances the security of a QKD implementation.

No extraneous parts or unused components will be contained in the system. Unused components are easier to overlook during security audits and can provide side-channel attacks into a system. Third-party tools and components that contain unused subcomponents may be allowed if approved for use on the system.

Data at rest must be encrypted by a FIPS-140-2-compliant quantum-resistant algorithm (Section 4.2.1) if it is categorized as sensitive or higher. Key storage for the algorithm is considered to be sensitive storage unless otherwise defined by a Security Classification Guide (SCG). Data modules that contain sensitive data should not be directly accessible nor reside on the same physical device or filesystem as the quantum communication components.

QKD2016.24. ***CAT II*** Unused Components

Discussion Unused components pose a security risk to the system. Unused components can contain sidechannel attacks, backdoors, and assets not reviewed for IA posture.

Check Contents Ask the system representative for a security/IA review of the system. Verify that component or code coverage statistics are included. Ask the system representative how unused components are found on the system. If component coverage statistics or methods to detect unused components do not exist, this is a finding.

False Positives Third-party components that are approved for use on the system may contain unused subcomponents that cannot be removed piecemeal from the third-party component.

IA Controls CM-6, CCI-000336

QKD2016.25. ***CAT I*** Data-At-Rest

Discussion Data-at-rest must be encrypted if it is sensitive or higher in classification or categorization. As a defense-in-depth approach to security, the compromise of a system should not also result in a compromise of the data it contains.

Check Contents Verify that all sensitive or higher data-at-rest is stored using a FIPS-140-2-compliant quantum-resistant algorithm. If it is not, this is a finding.

False Positives The Information Owner can define additional security procedures for data handling.

IA Controls SC-28, CCI-001199

QKD2016.26. ***CAT I*** Key Management

Discussion Keys to encrypt and decrypt data must not reside in the same location as the data the keys are protecting.

Check Contents Examine the system for private encryption keys or encryption metadata. Verify that it is not in the same location as the encrypted data.

False Positives Keys used and required as part of an application may reside with the application.

IA Controls SC-12 (2), CCI-002443

QKD2016.27. ***CAT II*** Temporary Storage

Discussion Temporary data used in the system must be cleared when it is no longer needed. Residual data is not always stored with its protection layers intact.

Check Contents Ask the system representative to demonstrate how data are cleared from buffers and purged from temporary locations after use. If data is still readable or obtainable at a temporary location, this is a finding.

IA Controls SC-4, CCI-001090

Sane Defaults

The system must be configured in a secure-by-default state. While the system may be able to support more than just its target installation needs, the additional features must be turned off by default.

QKD2016.28. ***CAT II*** Disabling of Unnecessary Features

Discussion Additional features and components within the system provide a larger attack surface. Disabling the unneeded features or removing them from the system prevents future compromise of those components from further compromising the rest of the system.

Check Contents Verify that all enabled functions are required for the system. If features or functions are enabled that are not needed for the functionality of the system, this is a finding.

IA Controls CM-7 (1), CCI-001762

Secure Failure

Failure scenarios must be taken into account. When compromise is detected or failures are encountered, the system must enter a safe state to avoid further compromise. Examples of failures include failing to decrypt the shared-key-encrypted message from one of the endpoints or detecting an eavesdropper by measuring Bell inequalities. In any scenario, the system should make sure that important data are not compromised.

QKD2016.29. ***CAT I*** Secure Failure Principle

Discussion The designer of the system will make sure that the Secure Failure Principle is followed. All data flow paths and system states should be accounted for.

Check Contents Ask the representative for data flow analysis on each possible path through the system. Verify that all scenarios are accounted for. If execution scenarios are

found that are not documented, this is a finding. If startup, shutdown, and intermediate states of the system are not defined, this is a finding.

IA Controls SA-4 (3), CCI-003107

QKD2016.30. ***CAT I*** Bell Inequalities

Discussion When quanta are received, the Bell Inequalities are calculated. If the inequalities have an error of >15%, an eavesdropper is assumed to be on the system and administrators are notified.

Check Contents Ask the designer for examples of how Bell Inequalities or some other eavesdropper detection method is implemented. If one is not implemented appropriately, this is a finding.

IA Controls SA-10 (1), CCI-000698

C.4.4 Cryptography

Quantum Key Distribution is usually intended to be used as a portion of a cryptographic system. With classical cryptography, Message Authentication Codes, hashes, and digital signatures could be checked for evidence of tampering. With QKD, Bell inequalities are added to the classical checking techniques to add eavesdroppers to the list of checked potentially pernicious parties. Classical data integrity mechanisms should still be employed with the chosen QKD method to provide a defense-in-depth solution.

FIPS

Assuming the recommendations in Section 4.2.1 have been implemented, the keys used in the QKD scheme should implement FIPS-140-2-compliant algorithms. The implementation of the

cryptographic scheme should follow FIPS-140-2 Annex A cryptographic module recommendations (`http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexa.pdf`). Key distribution should follow FIPS-171 guidance and protection.

QKD2016.31. ***CAT II*** FIPS-140-2 Cryptographic Module

Discussion If the QKD system does not implement the protections of a cryptographic module as defined in FIPS-140-2 Annex A, it may expose data inadvertently to attackers. Protecting data through all stages of the cryptographic process provides additional security against physical and side-channel attacks.

Check Contents Ask the designer if the design principles of FIPS-140-2 Annex A are followed. Choose pieces of the cryptographic process, such as key handling and storage, to verify that these design principles are followed.

IA Controls SC-13, CCI-002450

QKD2016.32. ***CAT II*** FIPS-140-2 Cryptographic Algorithm

Discussion The key being distributed by the QKD subsystem must implement a FIPS-140-2-compliant algorithm.

Check Contents Verify which encryption algorithm the key will be used for. If it is not a quantum-resistant FIPS-140-2-compliant algorithm, this is a finding. If the algorithm has known security threats, vulnerabilities, or other issues, this finding will be upgraded to CAT I.

IA Controls SC-13, CCI-002450

NSA

If the system handles classified data, verify that the QKD system is NSA-authorized as a Type 1 Encryption Device.

QKD2016.33. ***CAT I*** Type 1 Encryption

Discussion NSA-approved Type 1 encryption devices are authorized for the transmission of classified data.

Check Contents Verify that the system has been authorized as a NSA Type 1 encryption device. If it has not, this is a finding.

IA Controls SC-13, CCI-002450

Data Handling

Key management and data handling in memory are important defense-in-depth practices to keep an already compromised component from releasing even more sensitive data. Mitigation of potential successful attacks must be considered as part of the system.

QKD2016.34. ***CAT II*** Key Management and Data Storage

Discussion FIPS-171 key handling and storage requirements should be implemented. This prevents unauthorized distribution of keys and makes it more difficult for an already compromised system to be further compromised.

Check Contents Ask the system designer for evidence of FIPS-171 protections. At a minimum, verify that cryptographic records, logs, and metadata are encrypted using FIPS-140-2-compliant quantum-resistant cryptography.

IA Controls SC-13, CCI-002450

QKD2016.35. ***CAT II*** In-Memory Data Handling

Discussion Applications that handle keys should keep the data encrypted in memory to prevent TEMPEST-related attacks.

Check Contents Ask the system designer for evidence that data are properly encrypted in-memory. Ask the designer for the code that handles cryptographic keys in memory and verify that it is implemented as a protected data structure (e.g.: .Net *SecureString* rather than an unencrypted *String*).

IA Controls PE-19 (1), CCI-000993

QKD2016.36. ***CAT II*** Clearing of Sensitive Data Storage

Discussion The designer of the application must make sure that memory storing temporary or metadata information regarding the encryption scheme is overwritten or zeroed immediately after use.

Check Contents Ask the system designer for evidence that data are properly overwritten in-memory. Ask the designer for the code that handles sensitive data in memory and verify that the data are overwritten after use.

IA Controls PE-19 (1), CCI-000993

C.4.5 Auditing

Logging and auditing of events provides a defense-in-depth strategy for providing greater visibility to potentially malicious events on the system. Logs must be stored in a secure manner, must be reviewed regularly, and an automated alert when certain malicious events are triggered must be implemented. There are two important divisions of the Auditing family of checks: Logging (the storage of relevant information) and Auditing (the actual review of events and logs).

Logging

Information about the activities of the system should contain enough details to be able to trace a potentially malicious event from start to finish. These logs must be protected in a way that prevents unauthorized disclosure of the logged information.

QKD2016.37. ***CAT II*** Central Time Server

Discussion The internal time of the system must be synchronized to a central location.

Check Contents Ask the designer for evidence of time synchronization on the logging portion of the system. If the logging portion of the system is not synchronized to a central location, this is a finding.

IA Controls AU-8 (1), CCI-000160

QKD2016.38. ***CAT III*** DoD-authoritative Central Time Server

Discussion The internal time of the system must be synchronized to a DoD-authoritative time provider.

Check Contents Ask the designer for evidence of time synchronization on the logging portion of the system. If the logging portion of the system is not synchronized to a central location, this is a finding. If the system is synchronizing to a central location but not to a DoD-authoritative time source this is a finding.

IA Controls AU-8 (1), CCI-001492

QKD2016.39. ***CAT II*** Proper Datestamps and Timestamps of Logged Information

Discussion To be able to put together the events of a potentially malicious event, enough information of when the event took place must be stored.

Check Contents Ask the system designer for example log files. Review the logs and determine if, at a minimum, each event has stored with it a date and time that reflect:

1. year
2. month
3. day
4. hour, in a format that AM and PM can easily be determined
5. minute
6. second
7. timezone offset

ISO 8601 formatted date and time combinations are recommended. It is recommended (but not required) that the timestamp contains millisecond information.

IA Controls AU-12 (1), CCI-000174

QKD2016.40. ***CAT II*** Log files must have restricted access.

Discussion Log files can contain important information to an attacker. Preventing access to these log files from unauthorized sources can help prevent leaking additional information to a potential attacker.

Check Contents If the QKD system logs to a host containing an operating system covered by a specific operating system STIG, verify that the logfiles are included in the operating system's relevant logfile permission checks (EX: RedHat 6 STIG ID: RHEL-06-000135 Rule ID: SV-50424r2_rule Vuln ID: V-38623). If a specific operating system STIG does not apply to the logging device or the log files for the QKD system were not included in the log file check, verify that the permissions on the log file in a UNIX-based environment are at most 0600, or that the files in a Windows-based environment are only readable by the Auditors group and the service account that the logging subsystem is running under. If general users or administrators have read access to the log files, this is a finding.

IA Controls SI-11, CCI-001314

QKD2016.41. ***CAT II*** Log files must contain enough information to put together an event.

Discussion When required, the log must be reviewable to put together the specific actions of an event. For QKD systems, the authentication log, errors, and registrations must be stored.

Check Contents Ask the system designer for example log files. Review the logs and determine if, at a minimum, the following events are logged:

1. Authentication requests
2. Registration requests
3. Failed registrations
4. Failed authentications with the step in the authentication process the authentication failed at
5. Source and destination identifiers

IA Controls AU-12 (1), CCI-000174

Auditing

Reviewing the log files is just as important as logging the actual data. Monitoring of the logs provides faster notification of errors and potentially malicious events, and a robust auditing program can identify problems before they become catastrophes.

QKD2016.42. ***CAT II*** Logs must be sent to a central log server.

Discussion Remote logging to a centralized host provides a more robust monitoring of multiple subsystems. Data aggregation helps prevent and identify coordinated, distributed attacks. Centralized storage also guards against potential log tampering.

Check Contents Ask the system designer for access to the central log repository. Verify that the logs from the QKD system are stored by and pushed to the central log server.

IA Controls SI-4 (6), CCI-001265

QKD2016.43. ***CAT II*** Logs must be backed up.

Discussion Backing up the log files provides greater resilience against tampering and a central location in which to obtain historical data.

Check Contents Verify that logs are stored onto a separate server at least once every seven days. This can be accomplished by the central backup STIG check if the central server is not the same as the QKD system performing the logging.

IA Controls AU-9 (2), CCI-001348

QKD2016.44. ***CAT II*** Automated Log Review Tool

Discussion An automated central log server must scan for organizationally-defined potentially malicious events.

Check Contents Ask the system representative for a list of malicious events that are checked for. Verify that an automatic alert is sent to the correct role indicating when such malicious events are triggered. Verify that an alert is sent if there is a failure or warning state with the log system (such as insufficient disk space).

IA Controls AU-5, CCI-000139

C.5 Testing

The system testing process is vital for identifying bugs that can become large security incidents. Development of the system requires testing for security flaws, and documentation of those tests must be logged. The main actor and responsible party for these checks is the Test Manager.

QKD2016.45. ***CAT III*** Dedicated Security Tester

Discussion The Testing Manager must make sure that someone has been assigned to check for security defects. These security defects must include eavesdropping attacks between each communication stage.

Check Contents Ask the Testing Manager for the person(s) assigned to test for security defects. Verify that test cases exist and are checked for during testing of the system pertaining to its security disposition.

IA Controls CA-2 (2), CCI-001582

QKD2016.46. ***CAT II*** Assess Changes for Security Impact

Discussion Changes to the system must be assessed for security or IA impact.

Check Contents Ask the Testing Manager for recent change requests. Verify that the requests have been analyzed for IA impact.

IA Controls CM-4, CCI-000333

C.5.1 Test Plan

The Test Manager is responsible for documentation, including the Test Plan. The Test Plan must have approval, remain updated, and be executed for each functional release of the QKD subsystem.

QKD2016.47. ***CAT II*** Test Plan Approval

Discussion The AO must sign off on the test plan.

Check Contents Ask the system representative for a copy of the Test Plan. Verify that it has been approved by the AO.

IA Controls CM-6, CCI-000366

QKD2016.48. ***CAT II*** Test Plan Execution

Discussion Each major revision or update to the QKD subsystem must include an execution of the Test Plan. This should include regression testing, IA testing, and integration testing.

Check Contents Ask the system representative for a log of the most recent Test Plan execution. Verify that it matches the latest revision of the QKD subsystem. Verify that the execution of the Test Plan has been within the past 180 days.

IA Controls MP-6 (2), CCI-001030

C.5.2 Fuzzing and Data Manipulation

Fuzz testing must be performed to test against potential buffer overflow attacks, injection attacks, and proper error handling.

QKD2016.49. ***CAT III*** Fuzz Testing

Discussion The Test Plan must contain fuzz testing procedures.

Check Contents Ask the system representative for a copy of the Test Plan. If the Test Plan does not exist or does not address fuzz testing, this is a finding.

IA Controls SA-11 (8), CCI-003196

QKD2016.50. ***CAT III*** Buffer Overflows

Discussion The Test Plan must contain testing for buffer overflows.

Check Contents Ask the system representative for a copy of the Test Plan. If the Test Plan does not exist or does not address buffer overflows, this is a finding.

IA Controls SA-11 (8), CCI-003196

QKD2016.51. ***CAT III*** Injection Attacks

Discussion The Test Plan must contain testing for injection attacks.

Check Contents Ask the system representative for a copy of the Test Plan. If the Test Plan does not exist or does not address injection attacks, this is a finding.

IA Controls SA-11 (8), CCI-003196

QKD2016.52. ***CAT II*** Fuzz Testing and Injection Attacks

Discussion Fuzz Testing allows the tester to explore possible failure scenarios within the data flow. When errors are detected, they should be handled gracefully without revealing internal data structures.

Check Contents Ask the system representative for evidence that fuzz testing has occurred. If fuzz testing has not been performed, this is a finding. Ask the system representative for evidence that injection and impersonation attacks have been tested for. If they have not, this is a finding. Send a registration request or message that exceeds the expected size. If the unexpected size is not handled gracefully or if the system displays information about internal data structures, this is a finding. Send random data to the QKD subsystem. If the random data are not handled gracefully or if the system displays information about internal data structures, this is a finding.

IA Controls SA-11 (8), CCI-003196

C.5.3 Eavesdropping

Impersonation and eavesdropping must be included in the Test Plan. Confidentiality and integrity of data are the paramount in a QKD scheme.

QKD2016.53. ***CAT II*** Impersonation and Eavesdropping Attacks

Discussion The Test Plan must contain test procedures to check for impersonation and eavesdropping attacks.

Check Contents Ask the system representative for a copy of the Test Plan and for the Data Flow Diagram. If either document do not exist, this is a finding. Identify how eavesdroppers or impersonation attacks could be performed between different endpoints in

the data flow diagram. Verify that they are tested for in the Test Plan. If they are not, this is a finding.

IA Controls SA-11 (8), CCI-003196

C.6 Deployment

Each deployment of the QKD subsystem needs to be documented, managed, and secured against potential attacks. All of the design, documentation, and testing build up to the actual implementation and management of it.

C.6.1 Workplace Security

System engineers must be managed in a way that promotes visibility and non-repudiation. Responsibilities of individual engineers should be clearly defined, and security checks should be followed to protect against insider threats. Physical security helps prevent against both outside and inside malicious actors.

QKD2016.54. ***CAT II*** Security Check Procedures

Discussion The ISSM will ensure that the following checks and logs thereof are being completed:

- End-of-Day Checks
- Unannounced Security Checks
- Unannounced Personnel/Physical Penetration Tests

Check Contents Ask the system representative for evidence of each of the following. If they are incomplete or do not exist, this is a finding.

- End-of-Day Checks (logged every workday)

- Unannounced Security Checks (within the past year)
- Unannounced Personnel/Physical Penetration Tests (within the last year)

IA Controls SA-11 (8), CCI-003196

C.6.2 Maintenance

Continuing maintenance on the system allows implementations of it to receive timely security updates. The QKD subsystem must create a comprehensive IAVM (Information Assurance Vulnerability Management) policy. Deployments of the QKD subsystem must be able to subscribe to update channels and alerts for the subsystem.

IAVM integration testing must be performed to make sure that operations are not impaired by updates, so the comprehensive policy must address feedback from fielded versions of the QKD subsystem.

The development team does not have to have an active team dedicated to the development of the QKD subsystem but should have the resources available to find, fix, and continuously check for flaws.